VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zpracování a kategorizace textů v přirozeném jazyce

Natural Text Processing and its Categorization

2020 Jan Kubica

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Kubica**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Zpracování a kategorizace textů v přirozeném jazyce
Natural Text Processing and its Categorization

Jazyk vypracování: čeština

Zásady pro vypracování:

V Pythonu jsou dostupné moduly pro zpracování přirozeného jazyka v angličtině, chybí ovšem podobné nástroje pro lemmatizaci a stematizaci pro češtinu. Jejich využití se nabízí při analýze dokumentů nejlépe pomocí netradičních algoritmů a výpočtů (například neuronové sítě). Cílem je připravit funkční prototyp pro zpracování textů v angličtině a češtině a jejich analýzu. Funkčnost demonstrujte na nejméně dvou datových sadách, jejichž doménu předem konzultujte s vedoucím práce.

Seznam doporučené odborné literatury:

[1] Morbus Iff, Tara Calishain (2003): Spidering Hacks
http://guidetodatamining.com/
[2] Maas, Andrew L., Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. "Learning word vectors for sentiment analysis." Proceedings of the 49th Annual Meeting of ACL: Human Language Technologies-Vol. 1. ACL, 2011. 142-150.
[3] Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations." HLT-NAACL. 2013. 746-751.
Miller, George A. "WordNet: a lexical database for English." Communications of the ACM 38.11 (1995): 39-41.
[4] Šajgalík, M., Barla, M., Bieliková, M.: Exploring Multidimensional Continuous Feature Space to Extract Relevant Words. In: Proc. of SLSP 2014, Springer-Verlag, 2014
[5] Zelinka I., Včelař F.: Evoluční výpočetní techniky. 534 stran, Ben 2008
[5] Zelinka I.: Umělá inteligence, neuronové sítě a genetické algoritmy. 126 stran, Vutium, 1998

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. RNDr. Petr Šaloun, Ph.D.**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 15. května 2020 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 15. května 2020 .....

Rád bych na tomto místě poděkoval doc. RNDr. Petru Šalounovi, Ph.D za spoustu jeho času, odborné vedení a mnoho rad při zpracovávání této bakalářské práce. Rovněž bych rád poděkoval své rodině a přítelkyni za velkou podporu a trpělivost.

Abstrakt

Zaměřením této práce byla problematika zpracovávání textu v přirozeném jazyce a jeho kategorizace. Konkrétním cílem bylo vyvinout program pro zpracování textů v češtině a angličtině a jejich následnou analýzu. Po zvážení výběru jazyka pro implementaci byl vybrán programovací jazyk Python a pro extrakci dat z internetu byla využita jeho knihovna Scrapy. Lemmatizace textů je realizována skrze knihovnu Majka. Program umí, po naučení z dodaných datasetů, porovnat několik možných algoritmů pro kategorizaci textu a nové data do daných kategorií zařadit. V programu je také implementováno shlukování textů pro kategorizaci bez počátečních datasetů.

Klíčová slova: Kategorizace textu, Scrapy, Python, Majka, web crawler, strojové učení

Abstract

The aim of this work was the issue of text processing in natural language and its categorization, and specifically to develop a program for processing texts in Czech and English and their subsequent analysis. After considering the choice of language for implementation was selected programming language Python and its Scrapy library was used to extract data from the Internet. Lemmatization of texts is realized through its library Majka. The program can, after learning from the supplied datasets, compare several possible algorithms for text categorization and include new data in the given categories. The program also implements grouping of texts for categorization without initial datasets.

Keywords: Text categorization, Scrapy, Python, Majka, web crawler, machine learning

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V současné době je internet ve světě využíván ve stále větší míře. Přístup k němu má téměř každá domácnost a mnozí z nás se na něj dostaneme skrze mobil, z naší kapsy. Otevírají se nám tak pomyslné dveře k obrovským příležitostem a výhodám, jaké si naši předci ani nedokázali představit.

Jednou z těch nejvýraznějších je bezpochyby dostupnost informací. Na internetu se nachází nespočetné množství uživatelských příspěvků, recenzí, zpravodajských a výukových článků a jiných textů, které společně tvoří zajímavý podklad pro zpracovávání těchto textů.

V této bakalářské práci bude popsán stručný úvod do problematiky automatizovaného sběru dat z internetu, strojového učení a algoritmů pro klasifikaci textu. Budou zde popsány použité nástroje, knihovny a již existující řešení pro klasifikaci textů. Práce zároveň obsahuje jednoduchý návod pro zacházení s vytvořeným programem a závěrem bude porovnání výsledků různých klasifikátorů a ukázka běhu programu na dvou případových studiích.

2 World Wide Web

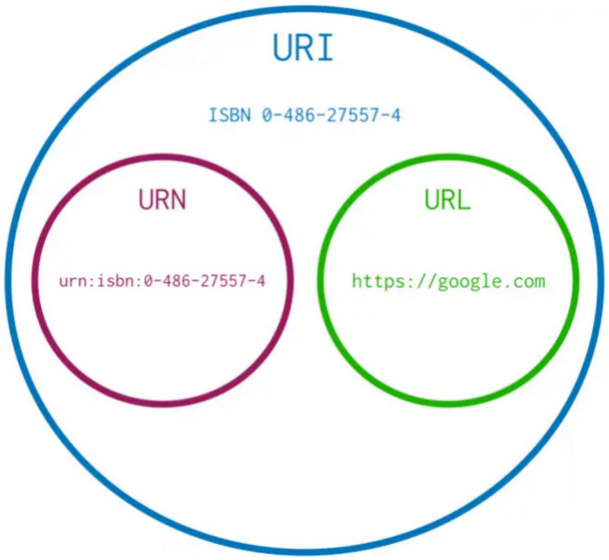
Samotná myšlenka o celosvětově propojené síti se začala uskutečňovat po roce 1985 [1], kdy se síť postupně začala šířit Evropou. Byl zaveden DNS (systém využívající URL, který převádí doménová jména webových stránek na IP adresy a naopak) a byly navrhnuty jazyky HTML a protokol HTTP, které se staly hlavními pilíři pro následné celosvětové používání webu. První webová stránka byla publikována 6. srpna 1991.

World Wide Web, hovorově označovaný také jako „web“ je nedílnou částí dnešního internetu. Ve své podstatě se jedná o systém s účelem zobrazování a sdílení dokumentů. Tyto dokumenty jsou uspořádány na webových stránkách, se kterými uživatelé mohou operovat pomocí webového prohlížeče. Z jednotlivých stránek je možná navigace na jiné webové stránky pomocí hypertextových odkazů. Ty se zapisují v URL formě (jako příklad můžeme uvést britský zpravodajský web `https://www.bbc.com`). Samotné webové stránky jsou vytvořeny pomocí jazyku HTML. V případě, že si chce uživatel určitou stránku zobrazit, začne zadáním URL adresy do prohlížeče. Prohlížeč pošle na server požadavek a ten mu vrátí načtenou stránku v souladu s HTTP protokolem, používaným na přenos dat a komunikaci mezi počítači.

V mezilidské komunikaci jsou běžně zmiňovány pojmy „internet“ a „web“ a jejich významy bývají často zaměňovány. Tyto názvy však označují dvě rozdílné věci. Internet je systém, který umožňuje propojení počítačové sítě okolo celého světa. World Wide Web je aplikace, která Internet využívá a společně s množstvím jiných služeb na něm poskytuje své služby. Od svých počátků na konci osmdesátých let se web značně rozrostl a je na něm ohromné množství informací, souborů a stránek, které jsou všechny navzájem propojených hypertextovými odkazy.

2.1 Internetové identifikátory

Uniform Resource Identifier neboli URI je identifikátor, textový řetězec, sloužící k upřesnění původu dat. Nejčastěji je používán pro použití řečených dat pomocí internetu. URI je popisován jako nejobecnější z identifikátorů a díky tomu je jeho základní formát velice svobodný. Jeho blízký příbuzný Uniform Resource Locator (URL) je speciální typ identifikátoru, který navíc k funkci URI specifikovat zdroj přidává způsob a cestu, jak se k původu dat dostat. Další důležitý identifikátor je Uniform Resource Name (URN), který cestu ke zdroji nebo jeho dostupnost vůbec neřeší a snaží se zdroj pouze identifikovat. Vztah mezi URL, URI a RUN je ukázán na obrázku 1 . [2]

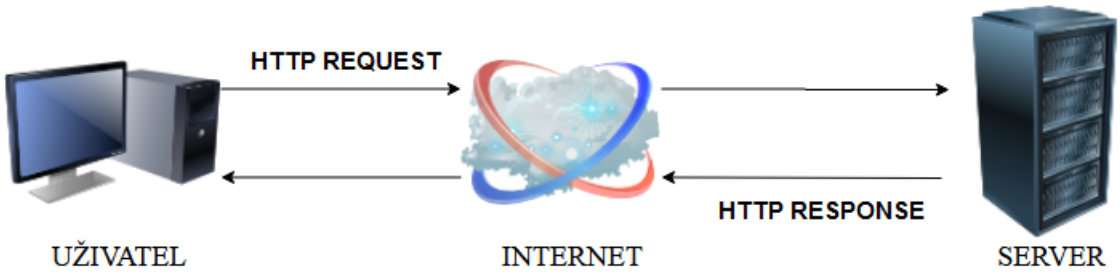


Obrázek 1: Vztah mezi URI, URN, URL [3]

2.2 Hypertext Transfer Protocol

Hypertext Transfer Protocol neboli HTTP je internetový protokol pro komunikaci s World Wide Web servery. Jeho úkol je pohyb informací z uživatelova počítače na server, nebo naopak. Také díky novodobé popularitě internetu a všudypřítomnému připojení patří HTTP mezi jedny z nejvíce používaných protokolů dnešní doby. V současnosti je ve spolupráci s dalšími protokoly jako např. FTP (protokol přímo delegovaný pro přenos souborů po internetu) a SMTP (protokol určený pro přenos zpráv elektronické pošty) používán pro přenos jakýchkoliv souborů. Jako identifikátor a specifikaci zdrojů po internetu využívá URL.

Samotný protokol pracuje formou dotazu (HTTP request) a odpovědi (HTTP response), ukázka na obrázku 2. Jakmile se uživatel připojí k internetu (nejčastěji internetových prohlížečem) a zadá příkaz k načtení dokumentu či webové stránky, odešle se serveru dotaz. Server pošle uživateli zpátky odpověď, ve které popíše stav dokumentu a za ní odešle data žádaného dokumentu. V případě, že bude uživatel chtít provést další akci, celý proces se opakuje nezávisle na předchozí výměně, ať už jde o stejný server, nebo ne. Tato vlastnost definuje HTTP protokol jako bezestavový protokol. To znamená, že neumí zachovávat stav komunikace, a proto může být pro implementace komplikovanějších nevhodný. Pro kompenzaci tohoto nedostatku se využívá dočasný session identifier [5], který identifikuje spojení serveru a uživatele. Uživateli je přiřazen session ID při prvním přihlášení na stránku a je udržován po dobu připojení. Session ID jsou obvykle krátkodobé a stávají se neplatnými jakmile je dosáhnut určitý cíl, jako dokončení nákupu či odhlášení ze stránky. Navíc byl HTTP rozšířen o HTTP cookies. Server si díky nim narozdíl od session ID dlouhodobě ukládá informace o spojení a mimo jiné ví, jestli byla stránka někdy v minulosti navštívena. [4]



Obrázek 2: Ukázka komunikace a použití HTTP protokolu

2.3 Hypertext Markup Language

HTML je jeden z hlavních jazyků používaných pro tvorbu webových stránek. Jeho vznik je úzce spojený s introdukcí a rozšířením World Wide Webu na začátku devadesátých let minulého století. Jazyk v jeho počátečních verzích nepodporoval grafické rozhraní, což byla vlastnost později doplněna do rychle se vyvíjejících nových verzí. HTML je využíván pro správné formátování textu a obrázků, za účelem korektního zobrazování v internetovém prohlížeči podle záměru.

Mezi jeho hlavní charakteristiky patří používání značek (tzv. tagů) a jejich dodatečných vlastností (tzv. atributů). Názvy tagů a jejich atributů jsou uzavírány mezi hranaté závorky. Atributy elementu mohou být jakkoliv upravovány nebo mohou být do elementu přidány další. Prvky (elementy), jsou tvořené otevíracím a ukončovacím tagem, čímž se definuje smysl vloženého textu mezi těmito tagy. Jako příklad může být uveden tag <b>, což je otevírací tag pro ztučnění textu. Počáteční značky mohou navíc obsahovat další důležité informace a vlastnosti, které je pomáhají dále definovat. Příkladem může být na internetu běžný odkaz, jehož vlastnost „href“ nám ukazuje destinaci, do které bude uživatel do kliknutí odkázán. Ukončovací značky žádné následné vlastnosti nenesou. Jako příklad můžeme uvést úryvek dokumentu, neboli výraz, který nás po kliknutí odkáže na stránky BBC. Ukázku kódu je možno vidět ve výpise 1.

```
<a href="https://www.bbc.com/news">Click here to go to BBC news.</a>
```

Výpis 1: Ukázka elementu s atributem

Značky můžeme z pohledu významu rozdělit na strukturální, popisné a stylistické. Strukturální značky udávají rozmístění objektů na stránce, mohou být vyjádřeny například značkou pro odstavec (<p>), nebo nadpisy (<h1>, <h2>). Značky popisné definují povahu obsahu objektu. Umožňují nám lehčí a efektivnější vyhledávání a automatizované zpracování dokumentů a webových stránek. Příkladem můžou být značky pro nadpis (<title>) nebo značky adresy (<address>), které můžeme vidět i v dokumentech XML. Posledním typem jsou značky stylistické, kterými je určován vzhled objektu při zobrazení na stránce. Typická stylistická značka je

(<b>), která se využívá pro zobrazení textu tučně. Tento typ značek v současnosti bývá nahrazován kaskádovými styly, které jsou lépe vybavené pro zobrazení textu i na alternativních zařízeních, jako jsou mobilní zařízení či tablety.

Celkový HTML dokument má předem určenou strukturu, v jaké musí být organizován. Jak je zobrazeno na výpise 2, dokument začíná deklarací typu dokumentu. Prohlížeč je díky ní informován o otevření HTML dokumentu a ví, jak stránku správně zobrazit. Následuje kořenový element (prvek html v ohraničených závorkách <html></html>), který reprezentuje začátek a konec dokumentu. Poté je sepsána hlavička (<head></head>), ve které definujeme autora, titulek a název dokumentu. Poslední částí dokumentu je jeho tělo (<body></body>), které zahrnuje vlastní obsah dokumentu a největší část stránky. [6]

```
<!DOCTYPE html>
   <html>
      <head>
         <title>Titulek</title>
      </head>
      <body>
         <h1>Nadpis</h1>
         <p>Toto je <a href="http://example.com/">odkaz</a> v odstavci.</p>
      </body>
   </html>
```

Výpis 2: Útržek HTML kódu

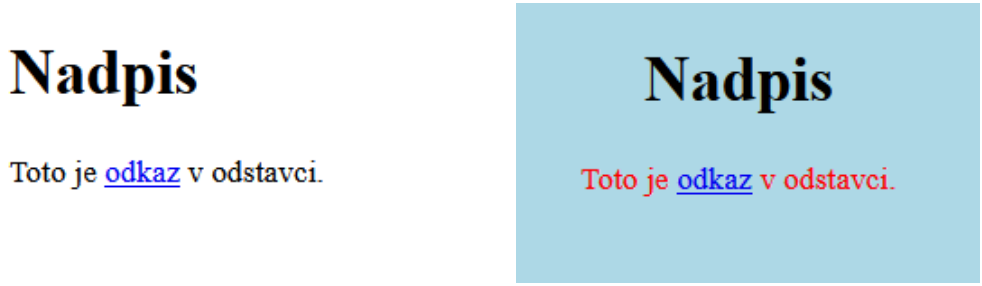
2.4 Cascading Style Sheet

CSS je jazyk, který úzce spolupracuje s jazykem HTML. Oproti HTML, jehož primární role je vytváření skutečného obsahu stránky, CSS určuje vzhled HTML dokumentu (rozdíl demonstrován na obrázku 3). To mimo jiné znamená volby barev, dekorační obrázky a umístění různých prvků. CSS soubory získávají funkčnost pouze po kombinaci s HTML souborem. Skladba CSS je složena z jednoho nebo více pravidel, jak je ukázáno na výpise 3. Pravidla zahrnují selektory (například „body“ pro výběr celého těla dokumentu nebo „p“ pro výběr všech odstavců) a bloky deklarací. Samotné deklarace se skládají z identifikátoru vlastnosti (například color, text-allign, padding), dvojtečky a hodnoty dané vlastnosti. Deklaraci ukončuje je středník. [7]

```
body {
  color: red;
  text-align: center;
  background-color: lightblue;
}
```

```
h1{
color: blue;
}
```

Výpis 3: Ukázka pravidla



(a) HTML bez CSS (b) HTML s využitím CSS

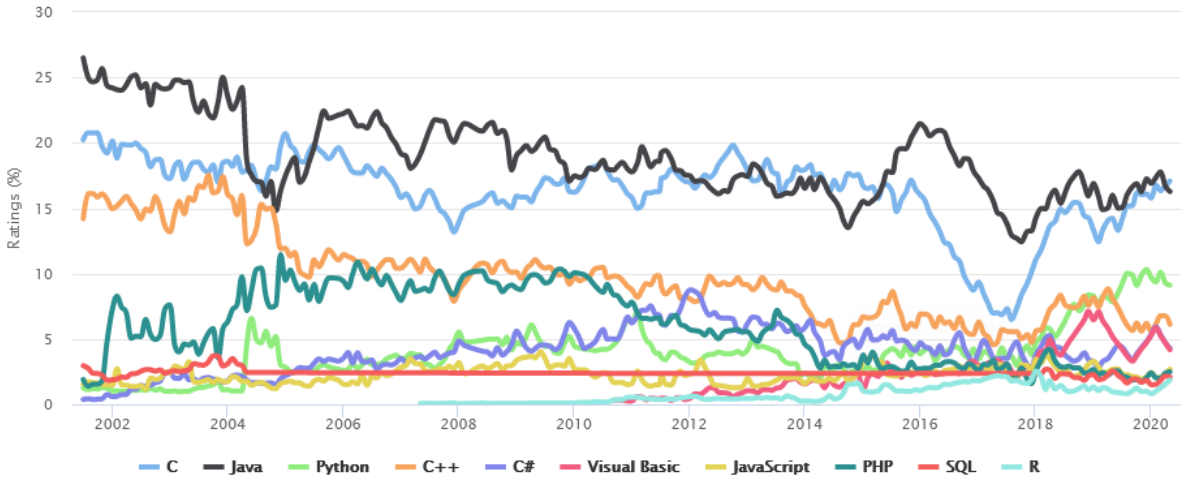
Obrázek 3: Ukázka použití HTML a CSS

3 Extrakce dat z webu

3.1 Python

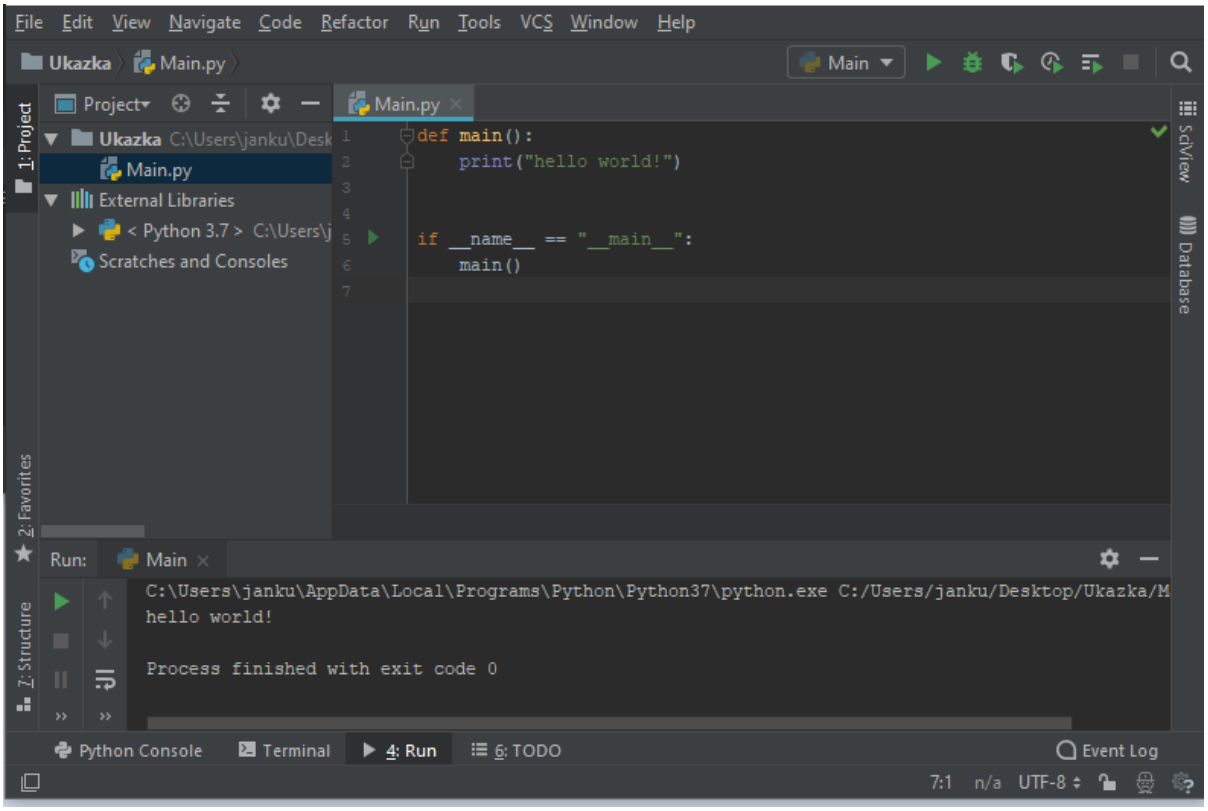
Python je skriptovací programovací jazyk, který nabízí dynamickou kontrolu datových typů. Nežádá se tedy specifikace datového typu u proměnných, výsledné programy jsou flexibilnější, přehlednější a obecnější. Stručnost zápisu a podpora jmenných prostorů nadále usnadňuje vývoj a rychlost psaní programů. Vysoká produktivita je dále navýšena dostupností a jednoduchou použitelností široké škály knihovních modulů, umožňujících snadné importování a řešení různorodé problematiky.

Python je jeden z nejoblíbenějších programovacích jazyků na celém světě, což lze vidět na obrázku 4. Je využíván ve velkém počtu populárních aplikacích, jako např. Dropbox, Instagram, Spotify nebo Facebook. Pythonovský program se snadno kombinuje s jinými aplikacemi, kde může sloužit jako jejich skriptovací jazyk, nebo cizí aplikace využít v roli pythonovského modulu. V této bakalářské práci jsou popsány a byly použity knihovny plně kompatibilní s jazykem Python. Knihovny jsou stáhnutelné z oficiálního, volně dostupného repozitáře balíčků s knihovnami, PyPI. [14]



Obrázek 4: Porovnání oblíbenosti programovacích jazyků [13]

Program napsaný v Pythonu ve svém výchozím nastavení pracuje skrze tlumočníka, který umí správně přečíst a vykonat kód. Otestování je možné například pomocí řádkového programu (CLI) do kterého napíšeme příkaz „python“. Otevře se nám tlumočník, do kterého můžeme psát příkazy, nebo spouštět kódy sepsané v jiných souborech. Pro lepší efektivitu a přehlednost práce je doporučeno využití IDE (anglicky „integrated development environment“). Příkladem může být open-source konzole PyCharm, která kombinuje textový editor, tlumočníka, průzkumníka souborů a další do jednoho programu. Ukázka konzole je zobrazena na obrázku 5.



Obrázek 5: Ukázka IDE pro Python

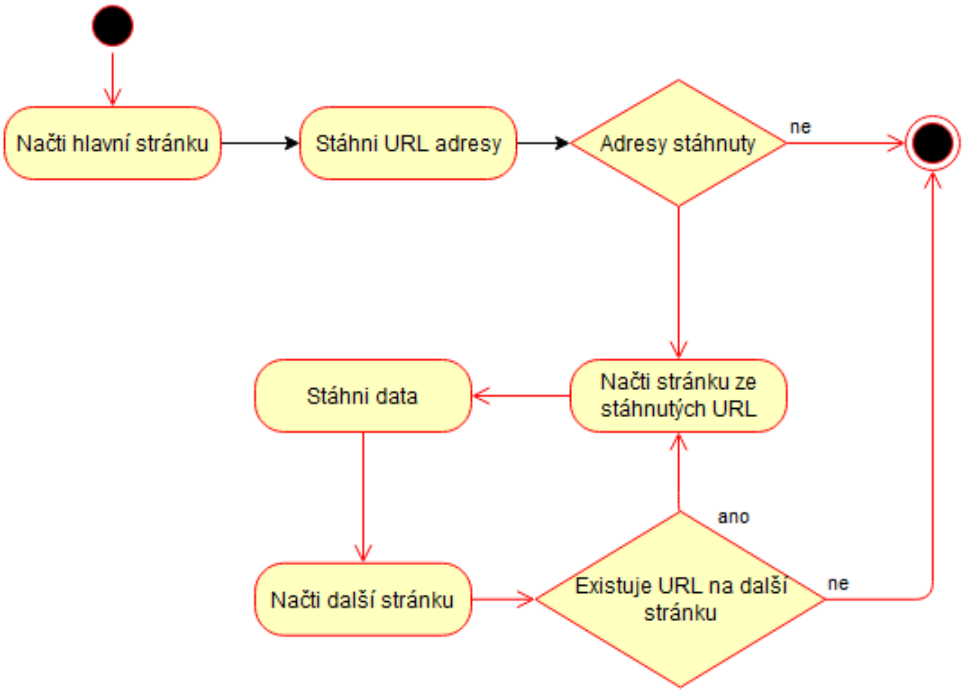
3.1.1 Pip Installs Packages

Je správce balíčků pro moduly Pythonu a jeden z esenciálních nástrojů každého Python programátora. Z tohoto důvodu většina moderních distribucí Pythonu přichází s pip předem nainstalovaným a nabízí jeho použití skrze příkazový řádek pokynem „pip install some-package-name“. Pip využívá Pypi jako hlavní repozitář a umožňuje z něj instalaci dodatečných knihoven či závislostí, které nejsou obsaženy v běžné knihovně. [15]

3.2 Web crawler

Web crawler je internetový bot [8], který má za úkol procházet webové stránky za účelem sběru dat a jejich případného ukládání do databáze. Diagram chování crawleru demonstrován na obrázku 6. Některé crawlery umožňují spuštění více procesů procházení a stahování najednou. Každý proces začíná s určeným seznamem URL adres, jejichž pořadí se určuje pomocí výběrové politiky daného crawleru. Po vybrání nejvýznamnějšího URL je navázáno spojení s DNS serverem a crawler je připojen na webovou stránku. Před stažením dat je z cíleného webu stažen soubor robots.txt, který definuje pravidla chování crawleru na stránce. Crawler poté začne analyzovat a ukládat si pro něj důležitá data. Těmi může být jejich obsah, metadata či změny v datech od poslední návštěvy. Důležité jsou informace o hypertextových odkazech, které jsou

na webových stránkách uživateli používány jako cesty na webové stránky jiné. Crawler odkazy identifikuje pomocí modulu na extrakci odkazů a přidá je do seznamu URL adres na postupné navštívení a zpracování. Každá z nalezených adres může být následně podrobena filtraci odkazů, jako je například černá listina. Pokud je adresa obsažena v této černé listině, dále se s ní nepracuje a je zahozena. Crawler zároveň porovnává, zda již byla adresa navštívena a jsou zkontrolovány i případné adresy duplicitní. URL adresy, které projdou kontrolou jsou podle priority seřazeny do seznamu URL adres k následnému navštívení. Chování web crawlerů je definováno kombinací vyhledávacích politik, nejdůležitější z nich jsou uvedeny v následujících podkapitolách. [9]



Obrázek 6: UML diagram chování web crawleru

3.2.1 Výběrová politika

Z důvodu obsáhlosti dnešního webu není pokrytí všech webových stránek reálné. Na tento fakt poukazuje i studie z roku 2009, která prokázala, že i největší vyhledávací systémy neobsahují více než 70 % [9] celkového webu. Z tohoto důvodu je důležité, aby crawler při stahování dat navštěvoval co nejvýznamnější webové stránky a z nich ukládal pouze nejpodstatnější data. Výpočet významnosti stránek je důležitým krokem ve sběru dat, bere se v potaz návštěvnost, URL, počet na stránku vedoucích odkazů, zpětných odkazů nebo hodnocení kvality stránky. Vývoj výběrové politiky musí být dynamický, jelikož se nikdy dopředu neví počet finálních stránek pro navštívení.

3.2.2 Politika opětovných návštěv

Politika opětovných návštěv určuje časové intervaly, v jakých crawler může opět navštívit určité stránky. Intervaly jsou významné pro zajištění účinného sběru a aktualizaci dat. Při jejich chybném určení a příliš krátkých intervalech nastává nepotřebné stahování již známých dat, v případě příliš dlouhých intervalů přichází k zastarání dat v naší databázi. Správná délka intervalů se kalkuluje pomocí různých faktorů, mezi které patří frekvence změn na stránce, výsledky algoritmů pro hodnocení významnosti stránky. V potaz se berou také technické vlastnosti serverů, aby se zamezilo zbytečnému přetěžování. Stáhnutí obsahu se vyjadřuje pomocí funkcí aktuálnosti stáří.

Aktuálnost (1) lze vyjádřit pomocí data poslední změny t v porovnání s datem stránky uloženém v databázi crawleru. Funkce se vyhodnotí jako 1, pokud je datum stránky shodné s datem v databázi. V ostatních případech je funkce vyhodnocena jako 0.

$$E_p(t) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{1}$$

Stáří stránky (2) vykazuje, zda jsou data stránky p uložené v databázi crawleru aktuální v čase t, či ne. Funkce je vyhodnocena jako 0 v případě, že p nebylo v čase t upraveno, v opačném případě se vyhodnotí jako čas změny t.

$$A_p(t) = \begin{pmatrix} 0 \\ t \end{pmatrix} \tag{2}$$

3.2.3 Politika zdvořilosti

Crawlery jsou v mnoha ohledech výkonnější a mnohem rychlejší než lidé samotní, v mnoha případech jediný crawler paralelně realizuje více procesů či stahuje rozměrné data. Na jedné webové stránce může být v jeden moment více takových crawlerů a servery, které nejsou na takový provoz vybaveny, mohou být zatěžovány a provoz na serveru zpomalován. Mezi nejčastější problémy, které jsou způsobeny crawlery patří přetížení serverů, narušení sítí, výpadky serverů a nadměrné využití síťových zdrojů. Pro snížení výskytu problémů se na web implementují zákazy nebo omezení vstupu botů do textového souboru robots.txt. Stanoví se, na které části nebo celé weby je pohyb crawlerů povolen a kam mají přístup zamezen.

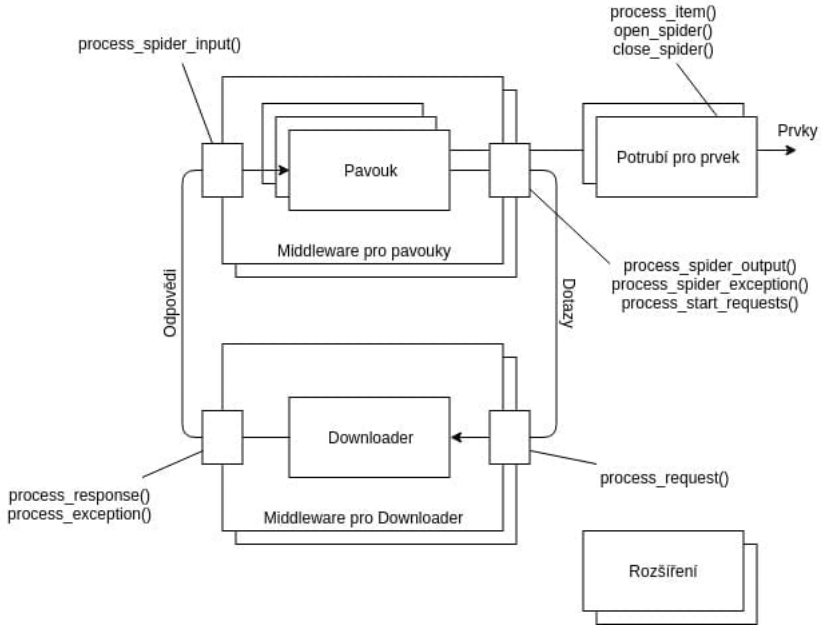
3.3 Scrapy

Scrapy je knihovna pro Python, přímo sloužící pro získávání dat z různých zdrojů. Vyniká zvláště při použití na stahování dat z internetových stránek a jejich následného formátování. Možnosti ukládání jsou ve formátu JSON, XML a CSV, nebo přímé skladování do databáze

pro pozdější offline prohlížení či provádění dodatečných činností. Pomocí knihovny Scrapy je umožněno kombinování dat z množství rozdílných zdrojů a jejich extrakce.

Architektura crawlerů s využitím knihovny Scrapy (ukázka zobrazena na obrázku 7) se v základu neliší od obecných crawlerů. Crawlery vytváří HTTP dotazy na webové stránky, kde pomocí CSS selektorů vybírají cílené atributy HTML elementů a jejich obsahy. Následné HTTP odpovědi zasílané do crawleru a jsou zpracovány a jsou vytvořeny jednotlivé prvky později ukládané do databáze. Crawler poté začne celý proces znovu vytvořením dalšího dotazu na jinou stránku.

Scrapy se navíc automaticky stará o cookies a tam, kde je to potřeba, i o přihlášení do webu. Každá část crawleru je implementovatelná do jiného projektu a není-li uživatel spokojen s výchozím naprogramováním, je možné funkcionalitu volně upravovat. Velká část těchto funkcí se nachází v middlewaru [10] (systém nabízející funkce nad rámec operačního systému) pro downloadery. Downloader je systém, který se stará o stahování dat z webů, při úpravě jeho implementace můžeme nastavit adresy na používání zabezpečených HTTPS protokolů, namísto nezabezpečených HTTP.



Obrázek 7: Architektura Scrapy [11]

Knihovna Scrapy úspěšně vzdoruje snížení výkonnosti kvůli pomalých nebo nepředvídatelných webových serverů, databází nebo jiných API (rozhraní pro programování aplikací) třetích stran. Je umožněn výkon mnoha paralelních operací, všech řízených z jednoho vlákna. To s sebou nese ohromnou výhodu při souběžném vykonávání dalších programů na stejném systému. Oproti vícevláknovým aplikacím není třeba synchronizace v kódu, což vede k dalšímu zjednodušení aplikace.

Scrapy je dlouhodobě vyvíjený a komunitně testovaný v mnoha projektech pro získávání dat z internetových stránek. Má nativní podporu CSS selektorů, důležitého mechanismu pro sběr dat z webů, představen samotnou Scrapy knihovnou. Tyto mechanismy se zaměřují na web, ze kterého následně vybírají určité části HTML dokumentů, specifikované pomocí CSS výrazů. Příklad implementace pavouka s počátečními URL adresami (seznam start_urls), načítáním nových odkazů (cyklus for a vyhledávání atributu "href"v HTML entitách) a jejich následné procházení je ukázán ve výpise 4. Zároveň je demonstrováno vytažení obsahů entit podle jejich atributů a ukládání do proměnné „title“. [8]

```
class BBCCrawler(scrapy.Spider):
  name = "BBC"
  start_urls = ['https://www.bbc.com/news']

  def parse(self, response):
      for href in response.css('div#news-top-stories-container a::attr(href)'
          ):
          yield response.follow(href, callback=self.parse)
      yield {
          'title': response.css('div.story-body h1::text').get()
      }
```

Výpis 4: Ukázka crawleru pro průchod zpravodajského serveru bbc.com/news

3.4 Extensible Markup Language

XML je obecný značkovací jazyk, který umožňuje snadné vytváření vlastních značkovacích jazyků pro různé účely a typy dat. Využívá se pro serializaci dat a je určen především pro datovou výměnu mezi programy či aplikacemi. V XML souborech není důležitý vzhled, ale funkčnost, nejsou přítomny informace o způsobu zobrazení. XML dokument neobsahuje předem definované značky, je proto na uživateli, aby si vymyslel značky vlastní. Tento fakt umožňuje přesnou a flexibilní definici každého dokumentu podle aktuální potřeby. Ukázka struktury XML souboru je na výpise 5. Nejvýraznějšími přednostmi XML je jednoduchost, mezinárodní podpora, nezávislost, malá velikost souboru a jednoduchý převod na jiné formáty. [12]

```xml
<?xml version="1.0" encoding="utf-8"?>
<items>
   <item>
      <name>BBC</name>
      <title>Slovenia cyclists hold anti-government protest</title>
      <date>9 May 2020</date>
      <text>Thousands of people cycled through Slovenia held protest.</text>
   </item>
</items>
```

Výpis 5: Ukázka XML souboru

4 Analýza textových dat

4.1 Lemma

„Lemma je reprezentativní slovníková podoba hesla, při automatickém zpracování jazyka je pak tato podoba v procesu lemmatizace přidělována každé formě v korpusu.“ [16]. Lemma u podstatných jmen je jejich první pád, příkladem mohou být slova „města“, „městům“ a „městy“, jejichž společným lemmatem je tvar „město“. U přídavných jmen se používá první pád jednotného čísla mužského rodu, u sloves je to infinitiv a u zájmen maskulinní tvar (tvary „ta“, „to“, „ti“, „tomu“ mají společné lemma „ten“). Lemma vzniká abstrakcí morfologických vlastností slovního tvaru. Spadají zde tedy slova se stejným kořenem, které se odlišují jen morfologickými afixy. Mezi afixy patří prefix (předpona) sufix (přípona, koncovka, postfix), infix (vpona), cirkumfix a interfix. Při automatickém zpracování jazyka morfologickými analyzátory se v průběhu lemmatizace lemma přidává ke každé slovní formě v korpusu.

Lemma je základní tvar daného slova a tedy i nositelem jeho lexikálního významu. Tuto skutečnost využívají a jsou podle ní budovány korpusově založené slovníky. V potaz musí ovšem být brána informace o samotném tvaru slova. Ten hluboce souvisí s významem, který mělo slovo v jeho původním znění nést. Lemma může být nadměrnou generalizací, která zanedbává závažné rozdíly mezi tvary slov. Jako příklad můžeme uvést lemma „nerv“, jehož význam může být interpretován jako součást těla - „poškozený nerv“, nebo jako součást kolokace „leze na nervy“. [16]

4.2 Lemmatizace

Proces, při kterém je slovo převedeno na jeho lemma. Lemmatizace je součástí automatické morfologické anotace a zároveň i desambiguace (neboli zjednoznačnění, jako je například jednoznačná interpretace slovního tvaru). Smyslem lemmatizace je identifikovat správný lexém a umožnit lidem následnou práci s původními slovními tvary i následnými lemmaty pro snadnější práci s korpusem. V případě, že lemmatizované slovo je nehononymní (jako například tvar „vytvoříme“, ze kterého nám vznikne lemma „vytvořit“), je Lemmatizace jednoduchý bezkontextový proces. V opačném případě je automatická lemmatizace komplikovaná. Pro korektní přiřazení se při tomto procesu musí brát v potaz kontext, ve kterém se dané slovo nachází.

Závažnou potíží pro automatizované i lingvistické lemmatizace jsou víceslovné spojení. Problémy zároveň vykazují i fráze. Ve frázi „nechat na holičkách“ není program schopen správně lemmatizovat tvar „holičkách“. Stejně tak při zdvořilé prosbě o uvolnění prostoru „Dovolíte?“ lemmatizátor nekorektně zařadí tvar jako lemma „dovolit“.

4.3 Lexém

Lexém je abstraktní jednotka vyjadřující množinu všech tvarů daného slova. Ve svém konkrétním tvaru je označován jako lex. Všechny lexémy se mohou skládat z množství morfémů, jako je kořen

a nebo předpona. V případě, že má dané slovo více než jeden tvar, k tomuto může dojít z důvodů jako skloňování nebo časování slov, jsou tyto tvary nazývány alolexy. Do slovníků se sepisuje jeho speciální reprezentativní tvar ve formě lemmatu. Jako příklad můžeme uvést lexém, který je reprezentován lemmatem „dům“. Lemma může být v textu v různých podobách, například „dům“, „domu“, „dome“, „domech“, „domy“. [17]

4.4 Morfologická značka

Tag, neboli morfologická značka je výsledkem automatické morfologické analýzy a desambiguace (jednoznačná interpretace) slovních tvarů v korpusu. Jeho význam spočívá v přenášení informací o daném slovu, a to vzhledem k jeho momentálním souvislostem. Tag je v českých korpusech tvořen soustavou značek, každá z nich pak nese význam dle pozice v kódu. Tyto značky jsou podřízeny souboru pravidel a hodnot, nazývaným tagset. Ten udává, která pozice nese jaké hodnoty. Příklad rozboru slova v určitém kontextu je společně s analýzou popsán v následující kapitole. [18]

4.5 Morfém

Morfém značí nejmenší smysluplnou část slova, která má gramatický či věcný význam. Nejčastěji se klasifikují na morfémy volné (kořeny slova), nebo morfémy vázané (afixy). Rozdíly mezi různými druhy kořenu a afixů jsou viditelné například na slově „strojopisný“, ukázáno na tabulce 1. [19]

strojopisný	kořen slova, který vyjadřuje základní význam slova
stroj**o**pisný	vpona, neboli morfém, který drží dvě části složeného slova dohromady
strojo**pis**ný	druhý kořen slova, odvozen od infinitivu slovesa “psát”
strojopis**n**ý	přípona převádějící slovo z podstatného na přídavné jméno
strojopisn**ý**	koncovka, která vyjadřuje pád a vztah slova k ostatním slovům ve větě

Tabulka 1: Rozdíly mezi kořeny a afixy

4.6 Segmentace

Segmentace je proces členění velkých částí textu (popřípadě textu celého) na menší úseky za účelem další analýzy. Úseky bývají rozděleny podle dle určitého typu nebo povahy. Nejčastější je segmentace větná, běžné jsou ovšem také segmentace na morfémy nebo grafémy. Rozklad na slova se nazývá tokenizace. Klasické využití segmentace je automatické, prováděné na počítači pomocí metod segmentátoru a bývá důležitou součástí samočinné analýzy korpusu. Segmentace je zpravidla netriviální, jelikož je nutné mimo jiné rozlišovat například interpunkční znaménka ukončující větu a zkratku. Z tohoto důvodu se konec věty pomocí značkovacího jazyka explicitně

vyznačí zvláštní značkou. Jako příklad značkovacího jazyka můžeme uvést jazyk XML. Názorný příklad si můžeme ukázat na následujícím textu:

„V samotných Litoměřicích žije přibližně 24 tis. obyvatel. Bývalo to lidnatější město.“

Pro korektní segmentaci a správné zapsání pomocí XML jazyka je text rozdělen do dvou vět. V potaz musíme brát tečku za tis., které jsou víceznačné a tudíž by zde program při netriviální segmentaci neuspěl. Zápis je zobrazen na výpise 6. [20]

```
<s>V samotných Litoměřicích žije přibližně 24 tis. obyvatel.</s>

<s>Bývalo to lidnatější město.</s>
```

Výpis 6: Ukázka segmentace textu

4.7 Morfologická analýza

Pod termínem morfologická analáza se v korpusové lingvistice odkazuje na počítačový program, neboli automatický proces analýzy textu. Při tomto procesu se ke každému slovu v daném textu přidají všechny jeho lemmata a tagy. Tyto tagy obsahují všechny morfologické informace o zvoleném slově, včetně jmenného rodu, čísla, slovního druh, osoby, pádu a vidu. V případě morfologicky homonymního slova mu je přidáno těchto tagů a lemmat více. [21]

Jako příklad uvedeme slovo „sním“ ve větě „Sním všechno jídlo.“. Morfologická analýza je ukázána v tabulce 2. Znak „-“ značí vynechání určení. Vysvětlení tagů se nachází v tabulce 3.

Původní slovo	lemma	tag
Sním	sníst	VpS–1d
	snít	VpS–1n

Tabulka 2: Příklad morfologické analýzy slova

1.	slovní druh: V – sloveso, P – zájmeno, N – substantivum, R – předložka, Z – interpunkce
2.	poddruh slovního druhu: p – prézens, P – osobní zájmeno, N – apelativum
3.	číslo: S – singulár, P – plurál
4.	jmenný rod: M – maskulinum životné, I – maskulinum neživotné. F – femininum, N – neutrum
5.	pád: 1 – nominativ, 2 – genitiv, 4 – akuzativ, 5 – vokativ
6.	osoba: 1 – první, 3 – třetí
7.	vid: d – dokonavý, n – nedokonavý

Tabulka 3: Význam tagů

4.8 Majka

Morfologický analyzátor Majka má podobu Python knihovny a CLI. Jeho účel je, jak název napovídá, analyzovat slova a podle vstupních parametrů vracet výsledky. Navazuje na svého předchůdce, morfologický analyzátor Ajka, se kterým sdílí podobné výsledky. Ajka byla vytvořena v rámci diplomové práce "Morfologický analyzátor češtiny" od Radka Sedláčka roku 1999[23]. Narozdíl od dnes již neaktualizovaného analyzátoru Ajka je Majka nově založena na konečných automatech a je tedy mnohem rychlejší a flexibilnější. Systém k danému slovu přidá jeho základní tvar, tag, všechna slova patřící ke stejnému lemmatu a všechna možná slova s diakritikou. [22]

Majka se používá jednoduše skrze CLI. Z adresáře, kde je majka uložena, zavoláme příkaz pro spuštění ve tvaru:

echo [analyzované_slovo] | majka -f [slovník]

Kde [analyzované_slovo] nahradíme naším vybraným slovem k analyzování a [slovník] vybraným slovníkem, podle kterého se bude slovo analyzovat. Finální příkaz společně s výstupem vypadá takto:

echo test | majka -f majka.w-lt

test:k1gInSc1
test:k1gInSc4
test:k1gMnSc1
testa:k1gFnPc2

5 Kategorizace textu

5.1 Strojové učení

Strojové učení [25] je věda, která se zabývá vytvářením algoritmů a technik, pomocí kterých je počítač následně schopný učit se z dat. Strojové učení je široce rozšířené do všech částí každodenního života, mnohdy nepozorovaně. Můžeme jej najít najít při rozpoznání řeči, překladu cizích jazyků, nebo ve zdravotnictví. Jako příklad strojového učení může být dán program pro rozpoznání dopravního značení. Ten se naučí dopravní značení rozlišovat pomocí ukázkových dat, které programu předložíme. Tyto příklady nazýváme trénovací data a jednotlivé příklady se označují jako trénovací instance nebo vzorek. Strojové učení je vhodné na robustní problematiky vyžadující vysokou úroveň ručního ladění, nebo obsahující dlouhé seznamy pravidel. Rovněž se používá pro prostředí, kde je často nutnost adaptace na nové data. Jeden algoritmus využívající strojové učení může výrazně ulehčit programátorskou práci a zároveň dosáhnout kvalitnějších výsledků. Žádná aplikace ovšem není bezchybná a výsledné výstupy mohou mít malé odchylky. Z tohoto důvodu je žádané algoritmy správně otestovat a vybrat algoritmus s největším procentem úspěšnosti. Rozlišujeme tři základní metody strojového učení, které jsou popsány v následujících podkapitolách. Každý proces strojového učení se taktéž dá rozdělit do určitých fází. [24]

Příprava vstupních dat: Zde program konvertuje data, které mu byla přidělena, do podoby, se kterými pak může pracovat. V případě klasifikátorů převede text na číselnou matrici, kterou dále analyzuje.

Trénink: V tomto kroku jsou programu poskytnuty trénovací data, podle kterých se vycvičí žádané funkcionalitě. Zde je využita jedna z metod strojového učení, které jsou popsány výše.

Testování a validace: Dáme systému za úkol provést dříve natrénovanou akci a poskytneme mu data, pro které máme předem definovaný správný výstup. Tímto krokem jsme po analýze úspěšnosti schopni vyhodnotit přesnost a rychlost daného programu. V případě, že se výsledek programu shoduje s předpokládaným výsledkem, můžeme systém využít pro práci s novými daty. Pokud výsledek podmínky nesplní, je nutno změnit trénovací algoritmus a proces zopakovat.

5.1.1 Unsupervised Learning

První metoda strojového učení je unsupervised Learning, neboli učení bez učitele. Učícímu se systému jsou poskytnuty pouze vstupní data, bez správných dat výstupních. Program ani neví, zda jsou příslušné data korektní a zda řešení existuje. Tato metoda je použita v prostředí, kde je nárok rychlé reakce na nepředvídatelné podněty. Zároveň se využívá pro prověřování známých procesů, které nevracejí předvídané výstupy.

5.1.2 Supervised Learning

Druhou metodou strojového učení je supervised Learning, neboli učení s učitelem. V tomto případě jsou programu předloženy vstupní data s předem vymezenými výstupy. Těmito výstupy bývají v případě klasifikačních algoritmů třídy nebo kategorie, v jiných případech mohou být použity i intervaly nebo hodnoty. Při použití této metody se nabízí možnost porovnání výsledků různých algoritmů a následně vybrání algoritmu pro nás nejvhodnějšího. Nevýhodou ovšem bývá použití v prostředí s vysoce proměnlivými daty a vyšší náročnost.

5.1.3 Reinforcement Learning

Reinforcement Learning, neboli zpětnovazební učení. Během učení a zpracovávání dat je systému dodávaná zpětná reakce, zda je jeho výsledek správný dle předpokladu, nebo ne. Pokud byl výstup hodnocen jako nežádoucí či nesprávný, systém upraví nebo kompletně zavrhne postup, kterým výsledek obdržel. V případě úspěšného výstupu je proces dále používán a zdokonalován. Toto učení je využito u průmyslových strojů, které se samy postupně zdokonalují a zpřesňují na základě prozkoumání jejich výsledků.

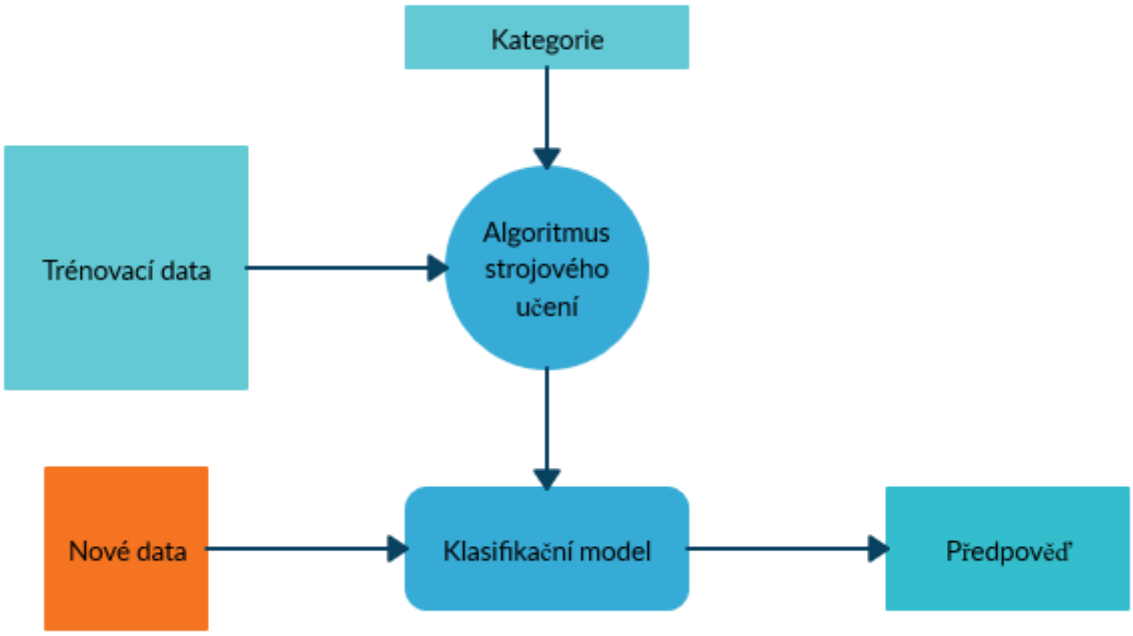
5.2 Klasifikace textu

Klasifikace textu (také kategorizace či tagování textu) je proces přiřazování předem definovaných kategorií k volnému textu. Textové klasifikátory jsou dále používány pro organizaci a strukturování textu. Jako příklad můžeme uvést rozřazování nových zpravodajských článků dle jejich témat, organizaci nových příspěvků na libovolném fóru podle jejich urgentnosti nebo rozlišování recenzí na pozitivní a negativní. Jako konkrétní příklad můžeme uvést následující větu:

„Americké prezidentské volby se letos uskuteční už v září.“

Klasifikátor tento text analyzuje a automaticky k němu přiřadí, do jaké předem definované kategorie spadá. V takovém případě záleží na uživatelském nastavení daných kategorií, jedny z možných se nabízí například „politika“ a „zahraničí“.

Klasifikace textů může být provedena samotným programátorem, který postupně interpretuje obsah každého z textů a ručně jej následně kategorizuje. Tento postup však je pracný a značně časově náročný, proto se častěji přistupuje k použití automatické klasifikace. Ta využívá strojové učení díky němuž je klasifikace mnohem efektivnější a rychlejší. Schéma klasifikátoru je ukázán na obrázku 8. [26]



Obrázek 8: Schéma klasifikátoru

5.3 Algoritmy klasifikace textů

V této podkapitole jsou popsány jedny z nejpoužívanějších algoritmů strojového učení pro klasifikace textů.

5.3.1 Naive Bayes

Naive Bayes je rodina pravděpodobnostních algoritmů, které umí předpovídat, z jaké pravděpodobnosti patří daný vzorek do každé z určitých kategorií. Tuto vlastnost můžeme při implementaci využít pro vyhledání nejvhodnější kategorie. Algoritmy jsou sestavené na základě Bayesovy věty, která udává spojitost podmíněné pravděpodobnosti jevu A a opačnou podmíněnou pravděpodobností, neboli jevem B. Bayesova věta se dá matematicky vyjádřit pomocí vzorečku 3.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3}$$

Kde P(A|B) je podmíněná pravděpodobnost jevu A za předpokladu, že nastal jev B. Zároveň musí platit výrok $P(B) > 0$. P (B|A) vyjadřuje pravděpodobnost jevu B, za předpokladu nastání jevu A.

Naive Bayes klasifikátory pracují s předpokladem, že mezi dvěma různými vlastnosti textu neexistuje závislost. Naive Bayes algoritmy vykazují nejlepší výkon při použití s malým vzorkem trénovacích dat (v řádů tisíců) a zároveň není vyžadován vysoký výpočetní výkon. Jejich využití

lze použít pro kategorizaci textu, kde se využívá četnost výskytu slov pro správné zařazení nebo například pro automatizaci lékařských diagnostik. [27]

5.3.2 Support vector machines

Support vector machines (SVM) jsou statistické metody využívající strojové učení s učitelem, s hlavním využitím v oblastech klasifikace a regresní analýzy (statistické metody pro odhadování hodnot náhodných veličin). Po přiřazení tréninkových dat je SVM rozdělí do dvou nadrovin. Trénovací algoritmus vytvoří model, který všechny další data přiřadí do jedné z nadrovin. Data patřící do rozdílných kategorií jsou reprezentovány jako objekty v prostoru, rozděleny viditelnou a co největší čarou. Jakékoliv další příklady projdou předpovědí a zařazením do kategorie podle toho, do jakého nadprostoru zapadnou. V případě, že nemáme předem připravené kategorie a učení s učitelem tedy není možné, využívá se text clustering (klastrování textů). Důležitou částí SVM metod je kernel transformation (jádrová transformace), během níž se nelineární data transformují na lineární, které dále můžeme zpracovávat a dělit na nadroviny. [28]

5.3.3 Deep learning

Deep learning, neboli hluboké učení, patří do skupiny postupů, které jsou inspirovány lidskou nervovou soustavou. Algoritmy hlubokého učení (dále jen AHU) se využívají například při zpracování přirozeného textu, automatické rozpoznání řeči či obrazu. AHU jsou především umělé neuronové sítě, jejichž základní a nejmenší prvek je neuron. Neurony jsou vzájemně propojeny a mezi sebou si vyměňují informace. Je však pravidlem, že samotný neuron smí mít pouze jeden výstup, který ale může být rozveden do více dalších neuronů. Vstupem může být libovolný počet informací z vnější, nebo z jiného neuronu.

Při příjmu dat z vnější si algoritmus vybuduje hierarchii, kde každý stupeň analyzuje a zpracovává data z vrstev předchozích. Opakování se provede tolikrát, dokud program nedojde ke správnému výsledku. Hodnotnou výhodou pro používání AHU je schopnost automatického vylepšování přesnosti predikce, narozdíl od tradičních algoritmů strojového učení, jejichž přesnost se po určitém množství trénovacích dat dále výrazněji nezlepšuje. K úspěšnému natrénování však AHU vyžadují mnohem větší množství dat, označené vzorky v řádů miliónů. [29]

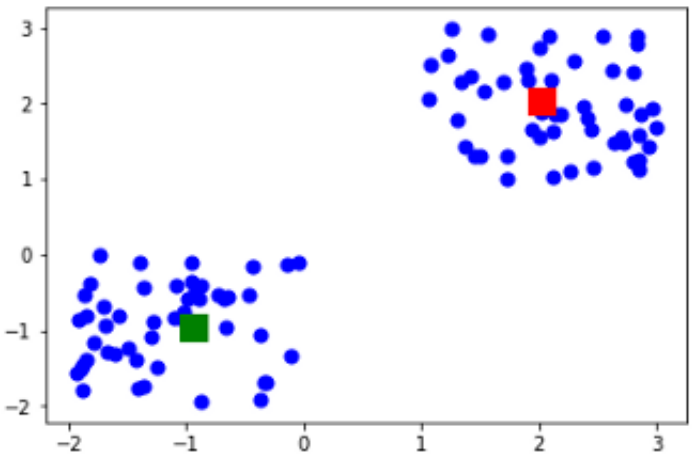
5.4 Shlukování textu

Text clustering, neboli shlukování textů je proces analýzy nestrukturovaných textů a jejich následné rozřazení do skupin. Stejně jako klasifikace se shlukování zabývá identifikací podobností analyzovaných vzorků. Narozdíl od klasifikace je ovšem shlukování využíváno při učení bez učitele, jelikož nezařazuje data do předem definovaných kategorií, nýbrž seskupuje vzorky do shluků založených na jejich společných podobnostech. Při použití algoritmu pro shlukování textů program vyjme z analyzovaných dat deskriptory (slova definující daný text) a analyzuje jejich

četnost. Poté spočítanou četnost porovná s četností v jiných dokumentech a na základě shody zařadí dokument k jemu podobným, nebo vytvoří shluk nový. [30]

5.4.1 K-Means

Jednou z možností pro implementaci shlukování textů je Kmeans clustering, což je metoda kvantifikace vektorů (zpracování signálů umožňující modelování funkcí pravděpodobnosti), která rozděluje n vzorků dat do k shluků. Shluky jsou reprezentovány centrálními vektory, které jsou využívány jako těžiště. Samotné rozdělování nových dat je odvozeno od blízkosti daných dat k těžišti všech shluků. Program poté přiřadí analyzované data do shluku nejbližšího. Kmeans algoritmy vyžadují předem specifikovaný počet shluků, což při nekorektnímu určení vede ke špatnému rozdělení hranic shluků. Shluky a těžiště jsou graficky vyjádřeny na obrázku 9. [31]



Obrázek 9: Shluky a jejich těžiště [32]

5.5 Term frequency - Inverse document frequency

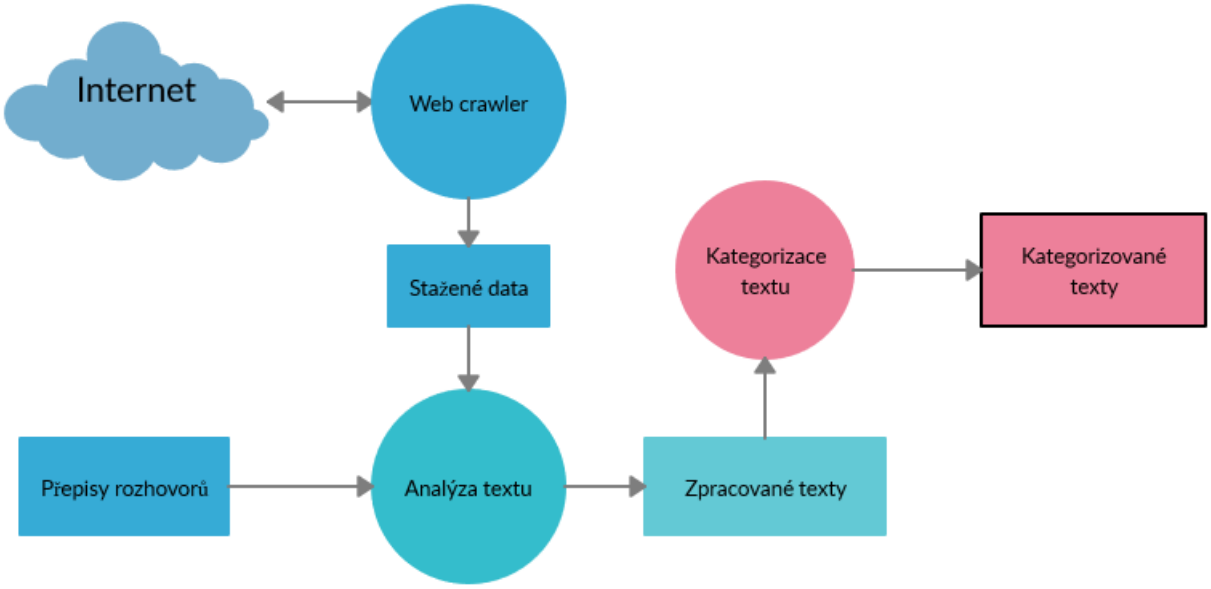
TF-IDF (Term frequency - Inverse document frequency) je metoda pro posouzení podstatnosti slov v analyzovaném textu vzhledem k ostatním textům a tudíž pomáhá vyvažovat obecně početné slova (jako jsou například slova být, mít nebo osobní zájmena) vůči slovům méně častým.

První část názvu - TF vyjadřuje výskyt daného slova ve vlastním dokumentu, vydělená počtem celkových slov pro zamezení nadhodnocování rozsáhlých korpusů.

IDF značí váhu slova v okruhu všech analyzovaných dokumentech. Čím roste jeho výskyt, tím méně váhy je mu připisováno. [33]

6 Vytváření aplikace pro kategorizaci textů

V této kapitole bude detailně popsán postup při řešení problematiky stahování webu pomocí web crawleru, zpracování a ukládání dat, datová analýza, následná klasifikace a shlukování pomocí příslušných nástrojů. Problematika je popsána na dvou případových studiích, první případová studie se orientuje na články z britského zpravodajského webu `https://www.bbc.com/news`. Druhá případová studie se týká fór webových stránek `https://www.cmp-radce.cz/community/`, které fungují jako komunitní rádce pro lidi s cévní mozkovou příhodou a jejich blízké. V projektu je zároveň řešena analýza lokálních záznamů rozhovorů s postiženými osobami.



Obrázek 10: Schéma aplikace

6.1 Výběr vhodného web crawleru

Při vybírání správného web crawleru byla otestována řada možností. Byly otestovány web crawlery s využitím uživatelského rozhraní, jako je například crawler Octoparse. Tento typ byl vyřazen zejména z důvodu zbytečné složitosti a nutnosti ovládání ze zmíněného uživatelském rozhraní. Crawlery s ovládáním přes příkazový řádek se osvědčily jako mnohem lepší alternativa, zejména z důvodu jednoduchosti užívání a kompatibility se zbytkem programu. Pro implementaci web crawleru pro stahování dat z internetu byly zváženy a otestovány Python knihovny, které jsou popsány v následující části. Text reprezentuje soupis poznámek, shrnutí a závěrů vyhodnocených v kontextu případových studí.

6.1.1 Mechanical soup

Mechanical soup je knihovna pro interakci s webovými stránkami, založená na dokumentaci knihovny BeautifulSoup. Její hlavní silou je simulace reálného člověka, vyplňování a odesílání

formulářů. Formuláře se ale nakonec projevily jako nepotřebné, nekompatibilita s Python verzí 3 nepřípustná a společně s nepodporováním s javascript stránkami byly tyto důvody dostačující pro vyřazení z možností. [34]

6.1.2 Beautiful soup

Beautiful soup je užitečný pomocný nástroj pro analyzování obsahu webových stránek a XML dokumentů. Je zároveň dobře dokumentovaný a je kompatibilní s nejnovějšími verzemi Pythonu. Beautiful soup ovšem dokumenty pouze analyzuje a pro stahování webových stránek potřebuje být doplněn o pomocné balíčky. Beautiful soup je dobrá volba pro malé projekty, ale z hlediska rychlosti, následné práce se staženými dokumenty a všestrannosti v rámci následování URL adres a stahování stránek se prokázaly lepší varianty. [35]

6.1.3 Lassie

Lassie je jednoduchá Python knihovna pro extrakci základních dat z webových stránek. Knihovna má jednoduché ovládání a přirozenou integraci do jakéhokoliv Python projektu, jevila se tak jako vhodná volba. Lassie byla zapojena do projektu a otestována se slibnými, ale bohužel nedostačujícími výsledky. Umí správně vybrat chtěné informace, je ovšem navržena pro vyjmutí jen základních dat, jako jsou určité fráze, nadpis a nebo stručný popis objektu. Je také limitována délkou výstupů a pro delší příspěvky tedy nevhodná. Nemožnost tuto skutečnost změnit vedla k následnému upuštění od této možnosti. [36]

6.1.4 Scrapy

Konečné řešení bylo realizováno pomocí komunitní knihovny Scrapy, která představuje plnohodnotné řešení procházení webu a ukládání dat, je umožněno přímé nastavení a řízení crawleru. Scrapy byl vložen do projektu, a i přes zprvu náročný skok do programování pavouka se projevil jako vhodná volba. Díky vlastní implementaci jsme schopni z mnoha stránek najednou vybírat pouze pro nás důležité informace a následně je ukládat do souboru.

6.1.5 Integrace do projektu

Crawler je nezbytnou součástí celého projektu, bez kterého by byl projekt nefunkční. Pro procházení webu jsou využity dvě třídy obsažené ve složce /spiders. Scrapy jednotlivé třídy využívá pro definování logiky stahovaných dat, počátečních URL adres a povolených domén.

Třída BBCSpider je implementována za účelem procházení webu `https://www.bbc.com/news`. Na stránce zprvu posbírá všechna URL vedoucí na články o novinkách ve světě, které následně prochází. Z jednotlivých stránek si ukládá datum publikace, nadpisy a samotný text. Po ukončení procházení se všechny data uloží do XML souboru BBC.xml a jsou předány k následné analýze.

Třída CMPForumSpider je nastavena na procházení komunitního fora na adrese `https://www.cmp-radce.cz/community/main-category/`. Podobně jako BBCSpider si z počáteční stránky načte všechny témata, které následně navštíví. O příspěvcích publikovaných v jednotlivých tématech si ukládá datum zveřejnění příspěvku, jméno uživatele, který příspěvek zveřejnil a název tématu. Následně se stáhnuté data ukládají do souboru CMP.xml. Třída je zobrazena ve výpise 7.

```
import scrapy


class CMPForumSpider(scrapy.Spider):
    name = "CMP"
    start_urls = ['https://www.cmp-radce.cz/community/main-category/']


    def parse(self, response):
        for a in response.css('div.wpforo-last-topics-list a'):
            yield response.follow(a, callback=self.parse)
        yield {
            'title': response.css('div.wpforo-main h1::text').get()
        }
        for post in response.css('div.wpforo-post'):
            yield {
                'name': post.css('div.wpf-author-nicename::text').get(),
                'date': post.css("div.wpforo-post-content-top span::text").get()
                    ,
                'text': post.css('div.wpforo-post-content p::text').getall(),
            }
```

Výpis 7: Ukázka implementace třídy CMPForumSpider

6.2 Analýza textu

Pro analýzu textu byl vybrán morfologický analyzátor Majka, jehož velkou výhodou jsou slovníky nejen pro český, ale i anglický jazyk. Majka funguje jako řádkový program a jeho vstup i výstup jsou kontrolovány skrze příkazový řádek. Pro zjednodušení ovládání byl na tuto bakalářskou práci využit modul „Python wrapper“, který po nainstalování umožňuje práci s analyzátorem z vývojového prostředí a plynulou práci s daty.

Analýza stáhnutých dat z internetu je zahájena programem automaticky. Program využitím třídy ProcessScraped načte data z XML souboru (podle konkrétního nastavení programu se vy-

bere BBC.xml nebo CMP.xml soubor). Články jsou podrobeny segmentaci a jednotlivé slova jsou připraveny k lemmatizaci (odstraní se přebytečné znaky jako pomlčky, tečky, otazníky, čárky a vykřičníky). Analyzátoru Majka se jednotlivá slova otagují a je vytvořen lexém. Program si z lexému vybere nejvhodnější lemma a to je následně uloženo do seznamu, který obsahuje všechny lemmata v textu. Analýza textu má čtyři výstupy ve formě dvou souborů a dvou složek. Do složky originalArticles/ se ukládají záznamy původních dat pro budoucí reference a možnou opětnou kategorizaci. Složka preparedForClassification/ obsahuje již zpracované texty. Do textového souboru countedWordsInArticle.txt jsou vpisovány konkrétní počty lemmat pro každý článek. Celý proces analýzy je zakončen vepsání záznamů analyzovaných článků do souboru analysed_articles.txt, podle kterého program zajišťuje analýzu pouze ještě nezpracovaných textů. Záznam je zapisován ve formátu [AUTOR]-[TÉMA]-[DATUM].

V programu je možné zvolit nastavení pro analýzu záznamů v rozhovoru. Pokud je tato možnost zvolena, program skrze třídu ProcessFiles konvertuje všechny dokumenty obsažené ve složce B_Rozhovory_prepis/ do souborů s příponou .txt pro snadnější zpracovávání. Poté následuje stematizace textu, lemmatizace slov pomocí majky a ukládání záznamů do složky.

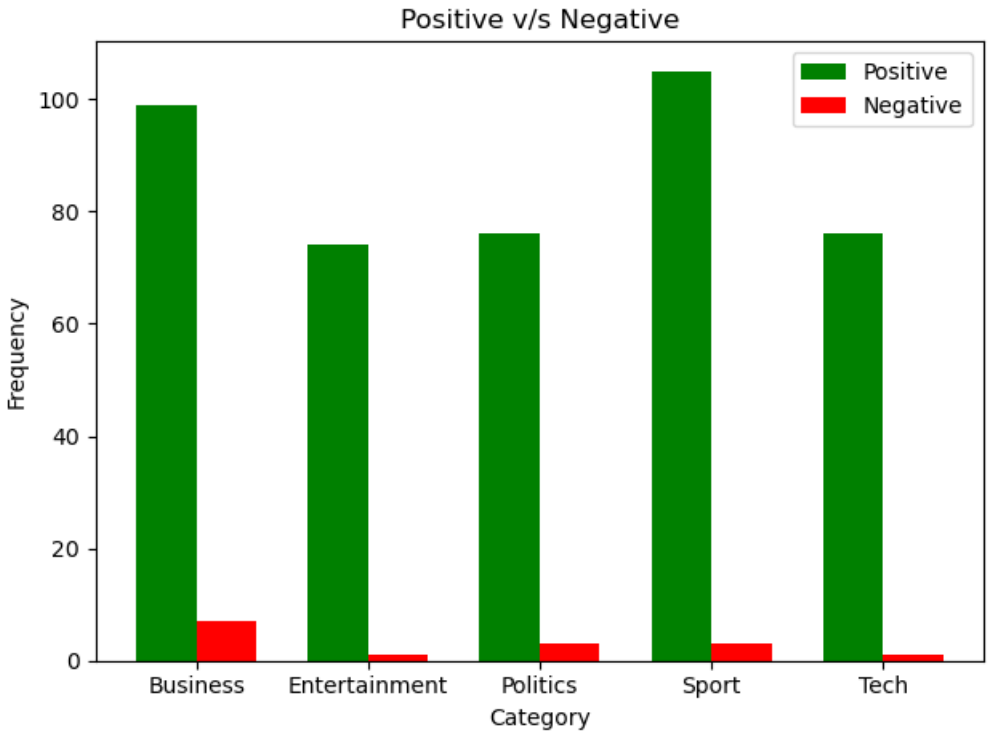
6.3 Kategorizace textu

Kategorizace textů se provádí použitím modelů klasifikátorů. Tyto modely jsou ovládány z třídy TextClassification. Ještě před tím, než je model schopen předpovídat umístění textů do kategorií, musí se zvolený model naučit své činnosti. Pro naučení jsou modelu předány data ve formě vybraných datasetů [37] v anglickém jazyce. Datasety jsou námi definované kategorie a v nich patřičně rozděleny ukázkové texty. Jako příklad uvedeme kategorii “sport”, která obsahuje 511 článků BBC o sportu. Datasety jsou načteny a rozděleny na trénovací a testovací data v poměru 80 : 20. To znamená, že 80 % dat z datasetu bude použito modelem pro naučení a zbylých 20 % nám zůstane na možné testování přesnosti.

Pro testování přesnosti byly použity klasifikátory vycházející ze SVM (SVC, SGDClassifier) a Naive Bayes (MultinomialNB) metod, kombinované s TF-IDF (TfidfVectorizer), nebo pouze jednoduchými vektory (CountVectorizer). Vektory se používají, jelikož modely narozdíl od lidí neumí zpracovávat textové soubory. Texty jsou převedeny na číselné vektory a tím je umožněno jejich další použití. Při testování model předpoví kategorii pro každý text z testovacích dat a výsledky jsou následně porovnány s výchozím datasetem. Zároveň je důležité, aby testovací data byla odlišná od dat trénovacích, testování by jinak nebylo objektivní. Porovnání přesnosti predikce můžeme vidět v tabulce 4 a na obrázku 11.

Klasifikátor	Vektor	Přesnost
SVC	TfidfVectorizer	0.96629
SGDClassifier	TfidfVectorizer	0.95618
MultinomialNB	CountVectorizer	0.95449
SVC	CountVectorizer	0.95441
MultinomialNB	TfidfVectorizer	0.95337
SGDClassifier	CountVectorizer	0.93988

Tabulka 4: Porovnání klasifikátorů



Obrázek 11: Porovnání korektních a nekorektních předpovědí

Proces trénování modelu pokaždé, co se program spustí je zbytečně zdlouhavý a nepotřebný. Model se musí natrénovat jenom jednou a poté se uloží. Při každém dalším spuštění se pouze načte ze souboru. Po načtení jsou modelu prezentována data pro klasifikaci ze složky preparedForClassification/. Model každý textový soubor projde a na základě jeho obsahu předpoví, do které z naučených kategorií patří. Výstup z celého procesu kategorizace je pro dva články je ukázán v tabulce 5, v programu je směrován do souboru classification_results.txt.

Jméno souboru	kategorie
BBC-Slovenia_cyclists_hold_anti_government_protest-9_May_2020.txt	politics
BBC-J_Crew_files_for_bankruptcy_protection-4_May_2020.txt	business
BBC-WU_promises_to_have_summer_tourist_season-13_May_2020.txt	politics
BBC-Mexico_receives_ventilator_shipment_from_US-6_May_2020.txt	tech
BBC-Baltimore_to_use_planes_to_patrol_city_May_2020.txt	tech
BBC-Tom_Cruise_Filming_in_space_and_other_stunts-6_May_2020.txt	entertainment
BBC-Why_is_there_US_backlash_to_masks_-5_May_2020.txt	politics

Tabulka 5: Výstup kategorizace

6.4 Shlukování textu

Můžeme říct, že modely s předem definovanými kategoriemi jsou vhodné pro kategorizaci textů, které mohou do úzké škály kategorií spadat. Jako příklad můžeme uvést příspěvky na zpravodajských stránkách. Pro všechny situace ovšem nemusí být datasety dostupné. V některých situacích může být proces učení na základě datasetů i nevhodný. Jako příklad můžeme uvést příspěvek, ve kterém uživatel napíše vtip. Model by tento příspěvek mohl nevhodně vyhodnotit na základě jeho obsahu. Z tohoto důvodu byl pro případovou studii na fóru a pro klasifikaci rozhovorů vybrán model, který využívá shlukování textu. Nezařazuje příspěvky do předem definovaných kategorií, ale seskupuje je podle podobnosti. Model zároveň vyřeší problém nedostatku datasetů, učí se z přidělovaných dat. Model je důležité průběžně přeučovat, aby byly jeho data a seskupení aktuální. Přeučování je obzvlášť důležité v počátečních fázích nasazení programu do běhu, jelikož vzorek textů pro tvorbu shluků může být malý a shluky nepřesné.

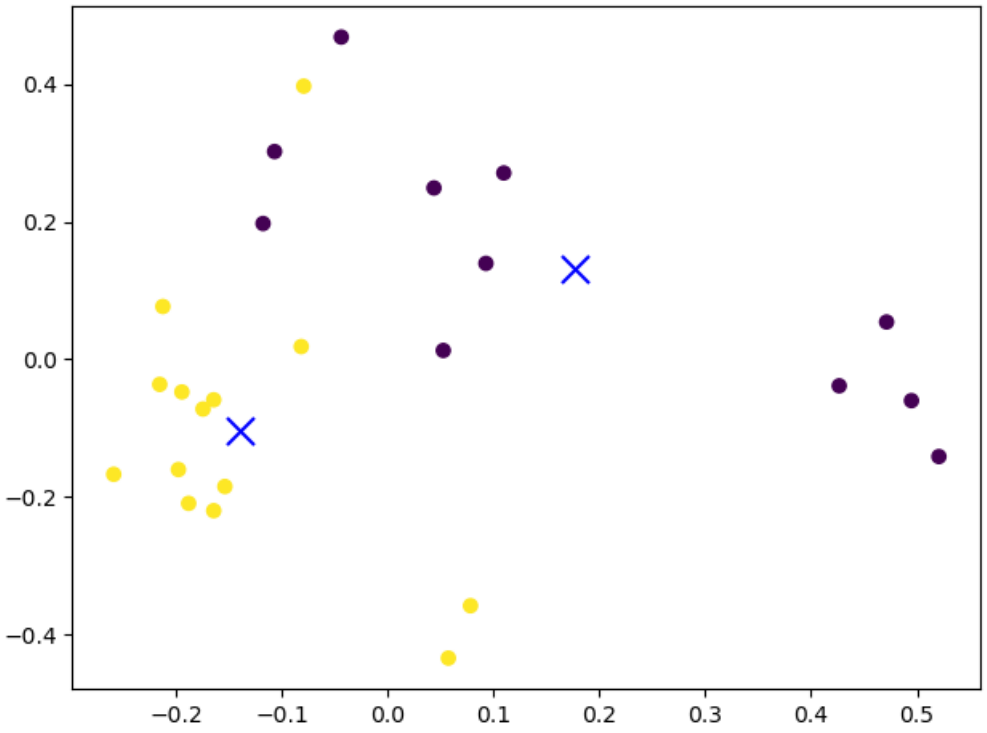
Pro shlukování textu využíváme algoritmus KMeans. Model při učení shlukování vyžaduje znát, kolik shluků má celkem vytvořit. Modelu je proto poslán seznam všech známých vzorků textu, na kterých je schopný předpovědět nejpravděpodobnější počet shluků. Poté je model naučen, shluky a jejich definující slova jsou vypsána (příklad v tabulce 6) a model je uložen do souboru, podobně jako modely klasifikátorů. Při použití pro klasifikaci nových příspěvků se model načte ze zmíněného souboru, vyfiltruje „stop slova“ (slova nevýznamné pro klasifikaci textu, jako například „ze“, „vždy“, „už“, „proč“) a přidělí příspěvky do adekvátních shluků. Výstup je směrován do souboru classification_results.txt, příklad je znázorněn v tabulce 7. Výstup po shlukování rozhovorů je zobrazen na obrázku 12. Výstup po shlukování příspěvků z fóra je zobrazen na obrázku 13.

Číslo shluku	Definující slova
0	vypadat, říct, zajít, dnes, výborně
1	strach, bolest, srdce, pochopení, účel
2	všechen, příšerný, vidět, hrozný, mít

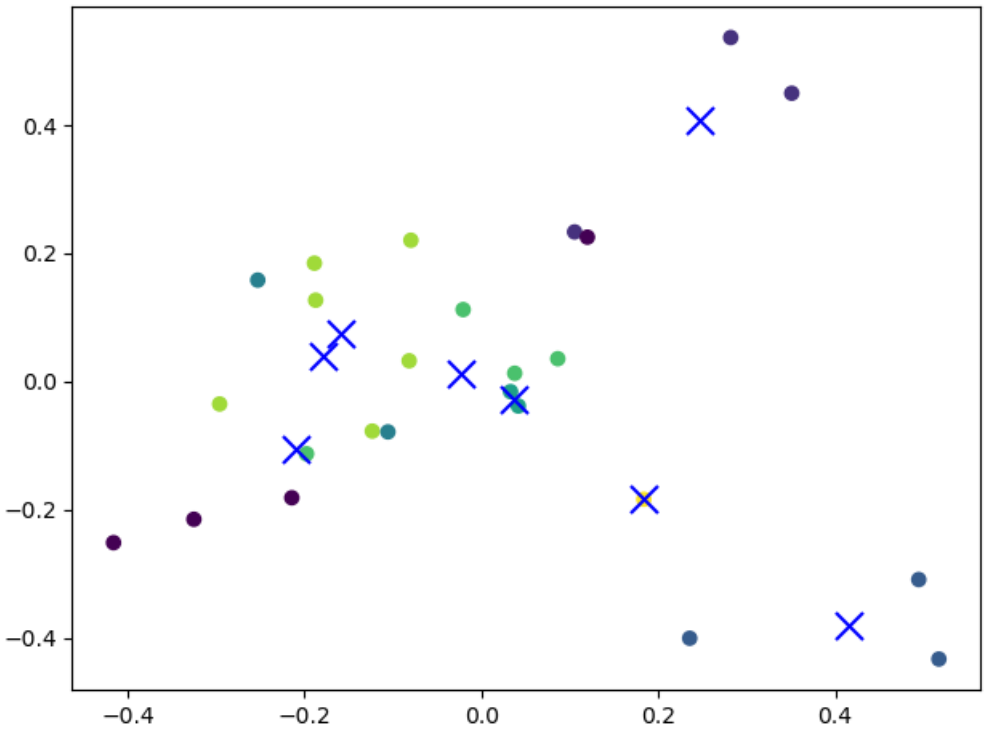
Tabulka 6: Ukázka shluků

Jméno souboru	Shluk
(@jankubica)-Radost_zivota-08_02_2020_2-07_pm.txt	[0]
(@marekvelky)-Vysoky_tlak-22_04_2020_11-13_am.txt	[0]
(@marekvelky)-Vysoky_tlak-22_04_2020_11-14_am.txt	[1]
(@petrpavel)-Nechodte_k_prochazkovi-07_05_2020_1-11_pm.txt	[2]

Tabulka 7: Ukázka výstupu po shlukování



Obrázek 12: Ukázka shlukování záznamů rozhovorů



Obrázek 13: Ukázka shlukování na datech z CMP fóra

7 Závěr

Cílem této bakalářské práce bylo připravit prototyp pro zpracovávání textů v anličtině a češtině a jejich analýzu. Funkčnost je demonstrována na dvou případových studiích. První studie je založena na extrakci dat a jejich zpracování v anglickém jazyce ze zpravodajského webu `https://www.bbc.com/news`. Druhá studie se týká analýze textu v českém jazyce ze záznamů rozhovorů a z fóra `https://www.cmp-radce.cz/community/`.

Pro správné pochopení webových stránek byly v druhé kapitole definovány pojmy a popsány základní jazyky, na kterých je dnešní web postaven. S webovými stránkami je úzce spojená kapitola druhá, ve které byla adresována problematika stahování dat z internetu, představen jazyk implementace vytvářeného programu Python a jeho knihovna Scrapy. Scrapy byla využita pro implementaci web crawleru pro průchod a stahování dat z již zmiňovaných webových stránek. Taktéž zde byl představen jazyk XML, pomocí kterého jsou data z crawleru ukládány.

Třetí kapitola je věnována definici pojmů a metodiky při analýze stahovaných textů a seznámení s analyzátorem Majka, který je následně v programu používán pro lemmatizaci českých i anglických slov.

Strojové učení učení bylo tématem čtvrté kapitoly, ve které byly mimo metody a problémy strojového učení taktéž přiblíženy již známé klasifikátory a jejich možnosti při práci s textem.

Praktické využití a detailní popsání předešlých kapitol bylo popsáno v kapitole poslední, kde jsme prošli celým vývojem prototypu aplikace. Byl diskutován výběr knihovny pro implementaci crawleru, charakterizován proces analýzy a byly porovnány kategorizátory. Modely kategorizátorů byly cvičeny a testovány na datasetu o velikosti 2207 souborů a pěti kategorií. Kategorie byly rozděleny na „business“, „entertainment“, „politics“, „sport“ a „technology“. Zároveň byla adresována problematika startu bez datasetů a navrhnuto řešení v podobě shlukování. Modely pro shlukování byly vycvičeny a otestovány na zápisech rozhovorů a příspěvcích na fóru CMP rádce.

Při zpracovávání této práce bylo použito strojové učení a kategorizace textů, jako součást datové analýzy. Na práci je možné dále navázat zpracováním a rozvinutím metod deep learning a metod automatického rozpoznání kategorií.

Literatura

[1] The Evolution of the World Wide Web [online]. 18 Water St, Colts Neck, NJ 07722: Monmouth Web Developers, 2018 [cit. 2020-05-08]. Dostupné z: https://web.archive.org/web/20181118231641/https://www.mwdwebsites.com/nj-web-design-world-wide-web.html

[2] URI Generic Syntax: URI, URL, and URN [online]. Washington, D.C.: The Internet Society, 2005 [cit. 2020-05-08]. Dostupné z: https://tools.ietf.org/html

[3] Url-uri-miessler-2019-white-e1576768407702. In: Danielmiessler [online]. San Francisco, California: Daniel Miessler, 2018, December 23, 2019 [cit. 2020-05-08]. Dostupné z: https://danielmiessler.com/images/url-uri-miessler-2019-white-e1576768407702.png.webp

[4] Hypertext Transfer Protocol – HTTP/1.1 [online]. Washington, D.C.: The Internet Society, 1999 [cit. 2020-05-08]. Dostupné z: https://tools.ietf.org/html/rfc2616

[5] What is a Session ID? [online]. 1000 De La Gauchetiere W Montreal, Quebec,: Springboard SEO, 2020 [cit. 2020-05-08]. Dostupné z: http://www.springboardseo.com/resources/what-is/session-id.html

[6] HTML Standard [online]. Cambridge, Massachusetts, United States: World Wide Web Consortium, 2020 [cit. 2020-05-08]. Dostupné z: https://html.spec.whatwg.org/multipage/

[7] HTML & CSS [online]. Cambridge, Massachusetts, United States: World Wide Web Consortium, 2019 [cit. 2020-05-09]. Dostupné z: https://www.w3.org/standards/webdesign/htmlcss#whatcss

[8] Představení knihovny Scrapy pro tvorbu web crawlerů [online]. Tomáš Kocman, 2019 [cit. 2020-05-09]. Dostupné z: https://www.root.cz/clanky/predstaveni-knihovny-scrapy-pro-tvorbu-web-crawleru/

[9] Complete Digital Marketing Guide Book for SEO, Social Media & Brand awareness: Definitive & Hidden Secrets of Digital Marketing to grow your business. Publicancy, 2019. ISBN 1794731881, 9781794731882.

[10] What is Middleware and How Does it Work? [online]. Martin Luenendonk, 2019 [cit. 2020-05-09]. Dostupné z: https://www.cleverism.com/what-is-middleware-and-how-does-it-work//

[11] Architektura Scrapy. In: Root.cz [online]. Tomáš Kocman, 2019 [cit. 2020-05-09]. Dostupné z: https://i.iinfo.cz/images/313/scrapy-2-prev.jpg

[12] Extensible Markup Language (XML) [online]. Cambridge, Massachusetts, United States: World Wide Web Consortium, 2016 [cit. 2020-05-09]. Dostupné z: https://www.w3.org/XML/

[13] TIOBE Programming Community Index [online]. Victory House II, Esp 401, 5633 AJ Eindhoven, The Netherlands: Tiobe [cit. 2020-05-14]. Dostupné z: https://www.tiobe.com/tiobe-index/

[14] What is Python? Executive Summary [online]. Wilmington, Delaware, USA: Python Software Foundation, 2020 [cit. 2020-05-09]. Dostupné z: https://www.python.org/doc/essays/blurb//

[15] What Is Pip? A Guide for New Pythonistas [online]. RealPython, 2020 [cit. 2020-05-09]. Dostupné z: https://realpython.com/what-is-pip//

[16] Pojmy:lemma [online]. Příručka ČNK, 2019 [cit. 2020-05-09]. Dostupné z: http://wiki.korpus.cz/doku.php?id=pojmy:lemma&rev=1551688187

[17] LEXÉM [online]. Brno: CzechEncy, 2017 [cit. 2020-05-09]. Dostupné z: https://www.czechency.org/slovnik/LEXÉM

[18] Seznamy:tagy [online]. Příručka ČNK, 2017 [cit. 2020-05-09]. Dostupné z: http://wiki.korpus.cz/doku.php?id=seznamy:tagy&rev=1497540816

[19] MORFÉM [online]. Brno: CzechEncy, 2017 [cit. 2020-05-09]. Dostupné z: https://www.czechency.org/slovnik/MORFÉM

[20] Pojmy:segmentace [online]. Příručka ČNK, 2014 [cit. 2020-05-09]. Dostupné z: http://wiki.korpus.cz/doku.php?id=pojmy:segmentace&rev=1416830148

[21] Pojmy:morfologicka_analyza [online]. Příručka ČNK, 2013 [cit. 2020-05-09]. Dostupné z: http://wiki.korpus.cz/doku.php?id=pojmy:morfologicka_analyza&rev=1379076709

[22] Pavel Šmerk. Fast Morphological Analysis of Czech. In Petr Sojka and Aleš Horák. Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2009. Brno : Masaryk University, 2007. p. 13–16. ISBN 978-80-210-5048-8.

[23] SEDLÁČEK, Radek. Morfologický analyzátor češtiny [online]. Brno, 1999 [cit. 2020-05-09]. Dostupné z: https://nlp.fi.muni.cz/projekty/ajka/ajka.pdf. Diplomová práce. Masarykova univerzita.

[24] Supervised and Unsupervised Machine Learning Algorithms [online]. BROWNLEE, 2016 [cit. 2020-05-09]. Dostupné z: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

[25] MITCHELL, Tom. Machine Learning [online]. McGraw-Hill, 1997 [cit. 2020-05-09]. ISBN 0070428077. Dostupné z: http://profsite.um.ac.ir/~monsefi/machine-learning/pdf/Machine-Learning-Tom-Mitchell.pdf

[26] Text Classification [online]. MonkeyLearn, 2020 [cit. 2020-05-09]. Dostupné z: https://monkeylearn.com/text-classification/

[27] A History of Bayes' Theorem [online]. LESSWRONG, 2011 [cit. 2020-05-09]. Dostupné z: https://www.lesswrong.com/posts/RTt59BtFLqQbsSiqd/a-history-of-bayes-theorem

[28] Support-Vector Networks [online]. The Netherlands: Kluwer Academic Publisher, 1995 [cit. 2020-05-09]. Dostupné z: http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf

[29] Deep Learning in Neural Networks: An Overview [online]. Schmidhuber, 2015 [cit. 2020-05-09]. Dostupné z: https://arxiv.org/pdf/1404.7828.pdf

[30] What is Text Clustering? [online]. Westborough, MA 01581i: insideBIGDATA, 2018 [cit. 2020-05-12]. Dostupné z: https://insidebigdata.com/2018/07/26/what-is-text-clustering/

[31] Understanding K-means Clustering in Machine Learning [online]. Garbade [cit. 2020-05-12]. Dostupné z: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[32] Kmeans clustering. In: Towards data science [online]. towards data science, 2018 [cit. 2020-05-09]. Dostupné z: https://miro.medium.com/max/348/0*vf08X44o0XgHVXvv

[33] Data Mining [online]. Stanford University, California: Cambridge University Press, 2012, s. 1–17 [cit. 2020-05-10]. ISBN 978-1-139-05845-2. Dostupné z: http://i.stanford.edu/~ullman/mmds/ch1.pdf

[34] Welcome to MechanicalSoup's documentation! [online]. Hickford, 2014 [cit. 2020-05-10]. Dostupné z: https://mechanicalsoup.readthedocs.io/en/stable/

[35] Beautiful Soup Documentation [online]. Richardson, L., 2007 [cit. 2020-05-10]. Dostupné z: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[36] Lassie [online]. Mike Helmick, 2014 [cit. 2020-05-10]. Dostupné z:https://lassie.readthedocs.io/en/latest/

[37] BBC Datasets [online]. Insight Resources, 2018 [cit. 2020-05-10]. Dostupné z: http://mlg.ucd.ie/datasets/bbc.html

Přílohy

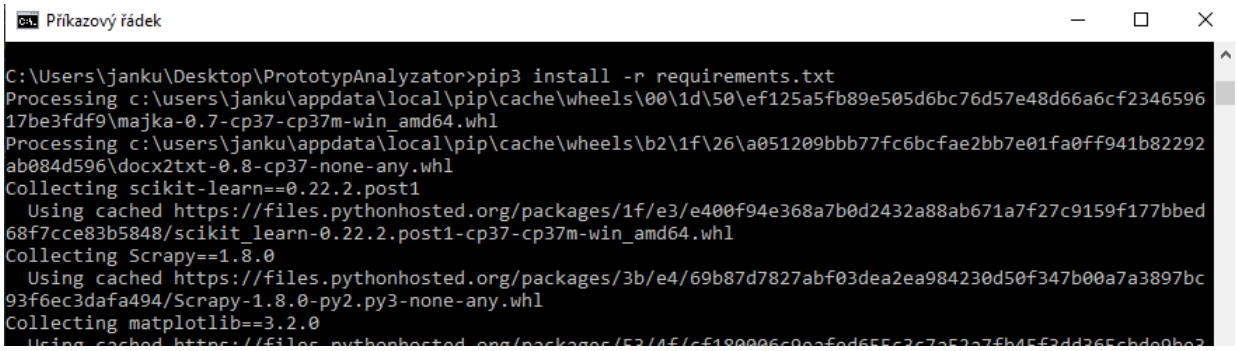
- Uživatelská příručka
- Elektronická verze práce
  - Datasety a kategorie
    - entertainment
    - sport
    - business
    - politics
    - tech
  - Výsledné modely
    - text_classifier_CMP
    - text_classifier_en
    - text_classifier_FILES
  - Slovníky
    - majka.l-wt
    - majka.lt-w
    - majka.w-lt
    - w-lt.en.fsa

Uživatelská příručka

Tato uživatelská příručka se vztahuje k softwaru vytvořenému v rámci bakalářské práce „Zpracování a kategorizace textů v přirozeném jazyce“, autor Jan Kubica. Práce byla předložena k obhajobě na jaře 2020 na VŠB - Technické univerzitě Ostrava, fakultě elektrotechniky a informatiky, katedře informatiky.

Program slouží ke stažení vybraných dat z internetu, jejich zpracování a analýzu a následnou klasifikaci do předem definovaných kategorií. Klasifikace je nastavena na stránky BBC. Program je zároveň vybaven algoritmem pro shlukování textu namísto klasifikace v případě, že nebudou dodány datasety. Shlukování textů je standardně nastaveno na web CMP a analýzu rozhovorů. Pro správný chod programu musí být splněny podmínky:

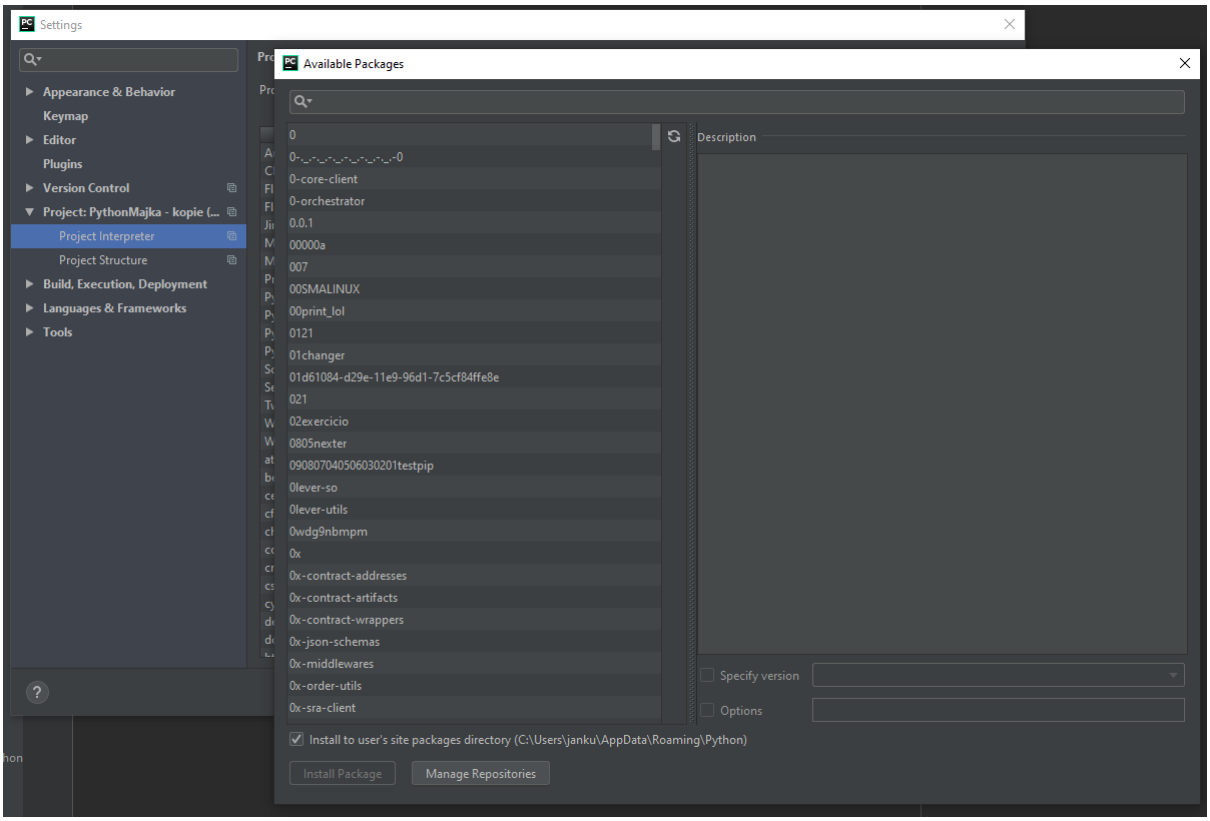
- Musí být nainstalován jazyk Python 3
- Pro stahování dat z webu musí mít počítač přístup k internetu
- Pro přidání a následnou analýzu nových záznamů rozhovorů je žádané jejich vložení do složky B_Rozhovory_prepis s příponou .docx nebo .txt
- Pro analýzu nových dokumentů je nutné dodržovat jedinečnost jejich názvů
- Pro správný analyzátoru Majka musí být nainstalováno C++ prostředí, například Visual Studio s C++;
- musí být nainstalovány a do projektu naimportovány knihovny majka, docx2txt, sklrean, scrapy, matplotlib. Pro automatické stažení a nainstalování je poteba v příkazovém řádku ze složky projektu zavolat příkaz **pip3 install -r requirement.txt**. Demonstrováno na obrázku 14.



Obrázek 14: Přidání modulů do programu pomocí requirements.txt

V případě, že se při prvotní inicializaci moduly pro projekt nenainstalují automaticky, je možné moduly nainstalovat pomocí příkazu -py -m pip install [MODUL]. Moduly ke stáhnutí

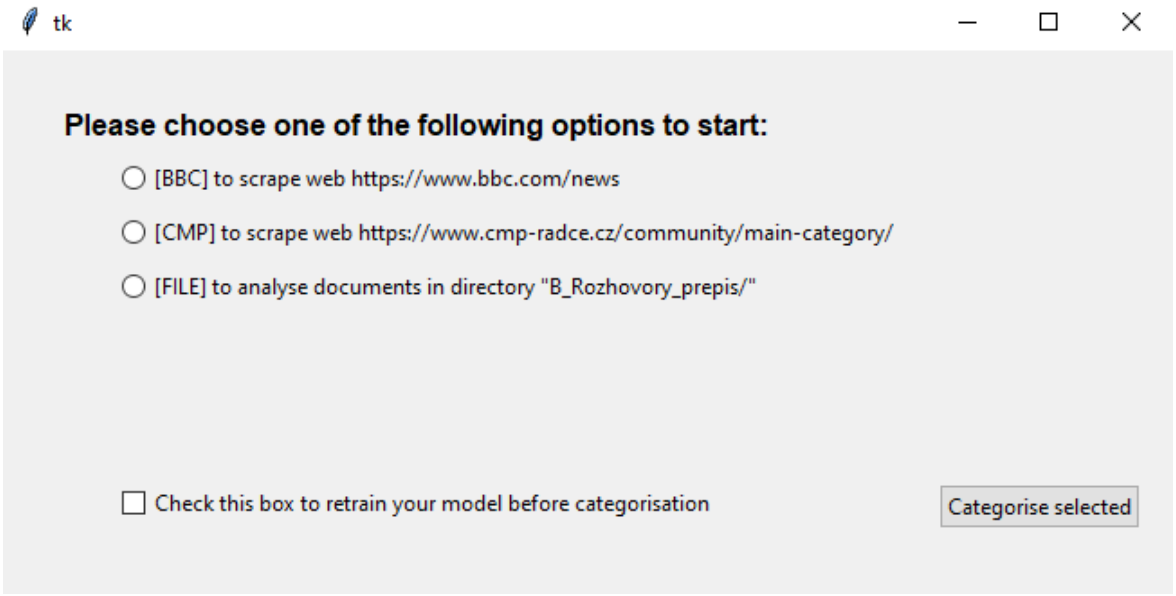
jsou: majka, docx2txt, sklrean, scrapy, matplotlib. Poté je nutné moduly do projektu vložit, pro IDE PyCharm je možnost vložení v File → Settings → Project Settings → Project Interpreter. Přidání nového modulu se provede pomocí znaku "+"° a vyhledání modulu podle jména, ukázáno na obrázku 15.



Obrázek 15: Manuální přidání modulů do programu

Spuštění programu

Program se spouští skrze třídu „Main“. Po spuštění je potřeba zadat, co má program udělat skrze grafické rozhraní, které je demonstrováno na obrázku 16. Uživatel si může vybrat ze tří možností. Program je nastaven pro stáhnutí webu `https://www.bbc.com/news` a následnou klasifikaci po výběru možnosti „BBC“, pro stáhnutí webu `https://www.cmp-radce.cz/community/main-category/` je nutno vybrat možnost „CMP“ a analýza rozhovorů se spustí po vybrání a odeslání možnosti „FILE“. Pro přetrénování modelu je nutné kliknout na check box. Vše se potvrdí kliknutím na tlačítko „Categorise selected“ a okno se zavře. Při přidání nových dat je doporučeno modely přetrénovat.



Obrázek 16: Start programu pomocí grafického rozhraní

Do programu byla implementována proměnná „use_gui“ ve třídě „Main“ pro případy, kdy není použití grafického rozhraní žádané. V těchto případech stačí změnit hodnotu proměnné na „False“ a program po spuštění grafické rozhraní vynechá. Uživatel bude vyzván k vybrání možností pomocí vstupu programu, jak je ukázáno na obrázku 17. Uživateli jsou představeny stejné možnosti, jako při výběru s grafickým rozhraním. Tentokrát však musí odpovědět slovně.

```
To run the program please insert one of the following options:
[BBC] for scraping web https://www.bbc.com/news
[CMP] for scraping web https://www.cmp-radce.cz/community/main-category/
[FILE] to analyse documents in directory "B_Rozhovory_prepis"
bbc
Do you want to retrain the model for this option? [Y] / [N]
Y
```

Obrázek 17: Start programu bez grafického rozhraní

Jakmile jsou příkazy odeslány, program začne automaticky vykonávat vybranou činnost. Program se ukončí samostatně a při úspěšném provedení operace se vypíše věta „Task was completed successfully.“. Uživatel může nalézt výsledky klasifikace a shlukování v souborech, které jsou rozdělené v tabulce 8.

Příkaz	**Soubor**
BBC	morphOutcomeFiles/scraperOutcomeFiles/BBC/classification_results.txt
CMP	morphOutcomeFiles/scraperOutcomeFiles/CMP/classification_results.txt
FILE	morphOutcomeFiles/processedFilesOutcome/classification_results.txt

Tabulka 8: Výsledky v souborech

Jako příklad zvolíme volbu „BBC“ a jelikož program znovu učit nemusíme, jako další volbu zvolíme 'n'. V tuto chvíli program začne pracovat a my počkáme na vypsání věty „Task was completed successfully.“. Jakmile program dokončí, přejdeme do textového souboru s cestou morphOutcomeFiles/scraperOutcomeFiles/BBC/classification_results.txt a podíváme se na výsledky.